Management of the Internet and Complex Services

European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable D9.1
Frameworks and Approaches for Autonomic Management of Fixed QoS-enabled and Ad Hoc Networks

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
International University Bremen, IUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universidat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, UniBwM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UniP, Romania

For more information on this document or the EMANICS Project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois — Campus scientifique
615, rue de Jardin Botanique — B.P. 101
F—54600 Villers Les Nancy Cedex
France

Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title: Frameworks and Approaches for Autonomic Management of Fixed QoS-enabled and ad hoc networks

Type: Public

Editor(s): George Pavlou, Apostolos Malatras, Antonis Hadjiantonis

E-mail: g.pavlou@surrey.ac.uk, a.malatras@surrey.ac.uk, a.hadjiantonis@surrey.ac.uk,

Author(s): Remi Badonnel, Mark Burgess, Markus Garschhammer, Panos Georgatsos, Stylianos Georgoulas, Antonis Hadjiantonis, Ralf König, Feng Liu, Emil Lupu, Apostolos Malatras, George Pavlou, Martin Serrano, Rolf Stadler, Ning Wang (in alphabetical order)

Doc ID: D9.1-v1.1.doc

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V0.1	November 03, 2006	Apostolos Malatras	First version, providing the ToC
V0.2	November 04, 2006	Mark Burgess, Rolf Stadler	Section 5 updated
V0.3	November 18, 2006	Martin Serrano, Apostolos Malatras	Section 3 updated
V0.4	December 05, 2006	Antonis Hadjiantonis, Remi Badonnel, Emil Lupu	Section 5 updated
V0.5	December 08, 2006	Apostolos Malatras	General editing, Sections 1, 2 and 7 updated
V0.6	December 19, 2006	Markus Garschhammer, Stylianos Georgoulas, Ralf König, Feng Liu, Ning Wang	Section 4 updated
V1.0	December 20, 2006	Apostolos Malatras	Final editing
V1.1	December 27, 2006	Antonis Hadjiantonis	Editorial corrections, inclusion of partners' comments

Legal Notices

Table of Contents

(This page is left blank intentionally)

1 Executive Summary

Autonomic Management has recently emerged as an evolution of automated management. In terms of management functionality, systems can be characterized as unmanaged, managed, predictive, adaptive, and autonomic. Adaptive closed-loop automated management is the first form of autonomic behaviour,with the ultimate target being fully distributed adaptive automated management, with relevant functionality built into the managed elements themselves. In this light, WP9 tries to integrate relevant know-how and approaches for both fixed/cellular and ad hoc/ubiquitous environments, answering questions and identifying key further research issues.

This work package is part of the Research Activity of EMANICS and will address issues of autonomic management in the context of both fixed Next Generation Networks (e.g. QoS-enabled) and ad-hoc/ubiquitous environments. The key objective of this work package and correspondingly of this deliverable, is to study and develop frameworks for the autonomic/automated management of both types of networks focusing on the required functionality that need to be present in such frameworks as well as the management technologies used to support the requirements of autonomic management of fixed and ad-hoc networks.

Autonomic management necessitates a multitude of functional building blocks to be achieved. It is a holistic view that enables the interaction of these building blocks in order to deliver management without supervision, in a proactive and intelligent manner. In this respect we address issues such as:

- We study the fundamental for autonomic management issue of context – awareness. We provide the foundations on how to use context information to achieve autonomy and classify context information according to various criteria. Approaches for efficient and descriptive context modelling and the corresponding storage mechanisms are also discussed.
- Section 4 presents an architecture for automated QoS management for fixed IP networks. We describe in brief the functional components that have to be involved in the process, ranging from the service negotiation phase to the service provisioning and deployment and we focus on the interface language/protocols involved in the automation of the negotiation phase and in the automation of the configuration of the QoS monitoring system of a provider.
- Frameworks and approaches for autonomic management of ubiquitous computing systems and ad hoc networks (as the major underlying network paradigm) are also extensively studied. We present research work on managing such systems and networks, detecting malicious behaviour, self-configuration and other issues regarding their autonomic management. Relevant experimental results are illustrated.
- Work on integrating the existing technology of cfengine with distributed management pattern algorithms is another aspect we address. Cfengine is an established framework for autonomic computing, used on up to a million computers around the world. Management patterns are an efficient way of signalling in overlay networks enabling distributed collaboration. The aim of this part of the WP9 work is to allow cfengine to benefit from management pattern technology without sacrificing its basic principles of autonomy. It serves as a case-study of providing an actual autonomic framework.

2 Introduction

Autonomic Management has recently emerged as an evolution of automated management. In terms of management functionality, systems can be characterized as unmanaged, managed, predictive, adaptive, and autonomic. Adaptive closed-loop automated management is the first form of autonomic behaviour, with the ultimate target being fully distributed adaptive automated management, with relevant functionality built into the managed elements themselves. In this light, WP9 tries to integrate relevant know-how and approaches for both fixed/cellular and ad hoc/ubiquitous environments, answering questions and identifying key further research issues. Relevant work is split into two parts: frameworks and technologies. Integration work in the area of frameworks addresses adaptive automated management approaches for fixed/cellular networks and distributed adaptive management for ad hoc / ubiquitous environments. Integration work in the area of technologies addresses policy-based management on one hand and emerging XML-based technologies and in particular Web Services as the unifying technology for future autonomic management systems. They complement well each other in the sense that they address various sub-problems from different perspectives. A key target will be the enumeration, harmonization and integration of existing approaches addressed by the various participants and the identification of key further issues. We focus on approaches and frameworks for autonomic management in this deliverable; enabling technologies and algorithms will be presented in our future work and more specifically in deliverable D9.2 of the EMANICS Network of Excellence.

The driving force behind the notion of autonomic management is to hinder the continuously growing barrier of complexity, which is intensified with the large scale, mobility and pervasiveness of modern computing systems and networks. Autonomic management refers to self-governing systems that are equipped to take decisions upon conditions in the absence of human administrators. The systems must be aware of their surroundings and possess a certain level of intelligence in order to be able to cope with emerging and dynamic events occurring in their realm. The paradox that is observed is that in order to be able to provide intelligence to current systems and networks to lessen their complexity, more complex systems have to be conceived and developed.

Autonomic management necessitates a multitude of functional building blocks to be achieved. It is a holistic view that enables the interaction of these building blocks in order to deliver management without supervision, in a proactive and intelligent manner. These building blocks include the following self-* properties:

- Self-awareness

  A system needs to have knowledge about itself in real-time so as to be able to identify the occurrence of certain events that refer to it. This can be extended to context-awareness regarding the environment a system operates in. It is the foundation of every autonomic management approach, since accurate and reliable context information guides autonomic decision making. In the absence of automatically and dynamically monitoring the system and its surroundings, autonomic management cannot be achieved.

- Self-configuration

  A system has to support self-configuration of itself or individual components under dynamic conditions so as to be able to adapt to changing environments. Self-

configuration supports the adjustment of the system's functionality in order to support higher level management decisions.

- Self-optimization

  High level optimization rules drive operation of systems that seek autonomic management. Systems and components continually seek ways to improve their behaviour and become more efficient, while the underlying monitoring scheme's significance is evident.

- Self-healing

  Systems can detect, diagnose, and repair hardware, software, and firmware problems upon detection. Potential problems can also be identified and actions to hinder them proactively can be employed on the system. Proper system operation under the desired functionality is thus ensured.

- Self-protecting

  Systems must be capable of detecting, identifying and protecting themselves against various types of security threats. This will assist in maintaining overall system security and integrity, while the operation of the system will not be influenced and will remain incessant.

Context-awareness is thus the foundation of every autonomic management framework and approach. The management of context information, by means of making it available to the system and properly modelling and processing it are extremely important tasks that need to be conceived upon design of any autonomic management solution. Context information serves as feedback and the enabler for every autonomic decision making.

Frameworks for fixed, QoS-enabled IP networks should allow for ease of administration and automated operation. Under this perspective, service planning, service provisioning, negotiation of QoS parameters and finally service enforcement should be designed and developed with the prospect of being automated. One should adopt a holistic view in providing automated/autonomic management for QoS-enabled networks; in parallel generic design is of paramount importance to allow for extensibility. Self-awareness through scalable monitoring is desired, while self-configuration according to monitored context information should not be disregarded.

Similar considerations need to be taken into account when building and deploying frameworks and approaches for ad hoc and ubiquitous environments. This emerging networking paradigm though brings a further set of requirements that also need to be handled. These include security concerns, such as misbehaviour of mobile nodes, scalability, support for node mobility and the dynamic nature of the environment, node configuration, heterogeneity of mobile devices and their limited resources, etc. Self-organization through clustering possibly is one important aspect, as well as self-protection in the sense of detecting malicious nodes and excluding them.

2.1 Purpose of the Document

This work package is part of the Research Activity of EMANICS and will address issues of autonomic management in the context of both fixed Next Generation Networks (e.g. QoS-enabled) and ad-hoc/ubiquitous environments. The key objective of this work package and correspondingly of this deliverable, is to study and develop frameworks for the autonomic/automated management of both types of networks focusing on the required functionality that need to be present in such frameworks as well as the

management technologies used to support the requirements of autonomic management of fixed and ad-hoc networks.

2.2 Document Outline

The document is organized as follows. Section 1 provides an executive summary of this deliverable. Section 2 serves as a general introduction, while it also presents the purpose of this deliverable and its outline. Principles of autonomic management are also presented. In Section 3 we study the fundamental for autonomic management issue of context –awareness. We provide the foundations on how to use context information to achieve autonomy and classify context information according to various criteria. Approaches for efficient and descriptive context modelling and the corresponding storage mechanisms are also discussed. Section 4 works towards the direction of automated QoS management and in this respect we present an architecture for automated service negotiation and service provisioning for achieving QoS in fixed IP networks. Frameworks and approaches for autonomic management of ubiquitous computing systems and ad hoc networks (as the major underlying network paradigm) is the focus of Section 5. We present research work on managing such systems and networks, detecting malicious behaviour, self-configuration and other issues regarding their autonomic management. Relevant experimental results are illustrated. The primary work of Section 6 is to integrate the existing technology of cfengine with distributed management pattern algorithms. Cfengine is an established framework for autonomic computing, used on up to a million computers around the world. Management patterns are an efficient way of signalling in overlay networks enabling distributed collaboration. The aim of this part of the WP9 work is to allow cfengine to benefit from management pattern technology without sacrificing its basic principles of autonomy. This Section serves as a case-study of providing an actual autonomic framework. Section 7 concludes this deliverable and discusses future directions.

3 Context-Awareness for Autonomic Management

Autonomic computing emerged as an initiative by IBM and has generated a very active research stream bringing together interdisciplinary domains. Autonomic computing refers to the self-managed operation of computing systems and networks, without the need for administrators but with high-level objectives dictating the system's functionality. The IBM autonomic computing blueprint [1] defines four distinct concepts behind autonomy, namely self-configuration, self-optimization, self-healing and self-protection [2]. The building block of all autonomic solutions is an autonomic element. This refers to the collection of one or more managed elements that are handled by an autonomic manager. The latter monitors the state of the elements, analyzes it and acting upon high-level objectives (typically defined as policies) imposes the execution of configuration changes on the managed elements. This process is repetitive [2][3].

It is evident that the foundation of any autonomic system is cross-layer context information. It is through this context information that the system will be able to be aware of its users' and its own state and surroundings and make it possible to adapt its behaviour accordingly. The notion of context-awareness makes a system minimally intrusive (e.g. in a given service with various options context awareness can reduce the number of user controls). An autonomic system can be seen as a human assistant that given user and environment context it will be responsible for making decisions in a proactive and dynamic fashion as well as anticipating user, application or network needs. In making decisions the assistant should not disturb the user except for an emergency.

Context information can be used to trigger cross-layer changes (network and application configurations) according to high level management rules, usually specified as polices through a policy-based management framework, leading to a degree of autonomic decision-making. Context information collected from all the nodes participating in the network refers to their computational and physical environment and is tightly coupled with the employed management system since it is this information being monitored that may trigger a certain need for (re-) configuration. Advanced techniques for efficiently and accurately modelling the context information collected by the nodes and its processing are necessary. A context model needs to be generic and extensible and also to cater for the accuracy of collected context information and its efficient processing. The need for generic design calls for a taxonomy of context information regarding autonomic computing systems, so as to have interoperable and standardized understanding of the useful context information. Issues regarding storage of context information are also extremely important.

The purpose of this section is to address the issues regarding context awareness for autonomic management of fixed IP and ad hoc networks. In this respect, it is crucial that we first agree on the scope and terminology of the key concepts in this evolving area of research. We put forward generic definitions as well as practical assumptions to map the notion of context for autonomic management as this has been conceived in EMANICS. The starting point for our analysis has been derived from the extensive literature survey that was conducted in order for the EMANICS Network of Excellence to comprehensively understand the current state of the art in autonomic communications. Based on the context-awareness literature survey and initial research findings, the context information definition, the taxonomy, the modelling and the storage issues are identified. Furthermore, we present our proposed solutions on tackling these issues.

3.1 Context-Awareness for Autonomic Management

In this section, we take a step back to ponder upon the terms used in the EMANICS Network of Excellence before revising and finalising a set of definitions that would ensure consistent discussion of ideas between the project partners as well as the wider research community. The emphasis is placed on the notion of context and context-awareness in regard to autonomic management of networks.

Context awareness has attracted the interest of a plethora of researchers from a variety of fields due to the apparent benefits obtained by allowing information from various sources to dynamically and adaptively influence the behaviour of an appropriately designed information system. There has been a plethora of definitions of the notions of context and context-awareness with none in particular emerging as the prevailing one. We do not distinguish significant differences between the various definitions given in the related literature and hence do not adopt one in particular. In previous studies context was defined in relation to its relevance to an application or service. While this is largely true, a more general definition should not include the usage (*i.e.*, whether for application or service) of context information since the term ***context*** could also be use for gathering of market and customer information or simply data mining exercises. In our view and based on our extensive and yearlong work on the subject we give the following description of the field of context awareness and the notion of context information

Definition 1

*The **context** of a system is the set of information of every nature that describes the system, influences system aspects and that is being affected by the system's operation, the ownership of which is not necessarily solely held by the system.*

Definition 2

***Context awareness** refers to the ability of a system to adapt dynamically and continuously its status and operation according to context information*.

Both of these definitions are generic and their applicability in the autonomic management of networks is straightforward.

In contrast, other kinds of definitions are more abstract. Between them, we can mention the definition expressed by Dey [4], who defines context as *any information that can be used to characterize the situation of an entity. An entity is defined as the person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.* In other words we can say that context information is any information that can be used by a service to deliver its functionality despite its nature.

The definition of *context* we have provided has been considered as the starting point for the definition and design of the proposed information model that will be used to represent the context information. This is because we consider that this definition reflects properly the heterogeneity that resides behind a concept like context. In order to clarify what context should express and how this should be done, it is important to understand what is context and what kind of information the model that would try to represent this concept should contain. So the first step is establishing a clear definition of context and classifying it into taxonomy.

The term ***service*** has been used rather extensively within the research community according to their corresponding application. First of all, it is important to agree upon the dichotomy between ***high-level services*** that are evident to the end-user (*e.g.*, video

conferencing, VoIP, web hosting) versus ***basic services*** that are diaphanous to the end-user (*e.g.*, routing, DNS). The former will rely on various forms of the latter, *i.e.*, for a high-level service to be properly deployed there is a need for the basic services to be operating effectively.

Definition 3

Context-aware service (CAS) *is a service offered by a group of computers (i.e., effectively a network) that can adapt to the dynamic nature of the environment. The network is fed by the pertinent context related to user and its environment that would enable the former to provide a comprehensive yet seamless service over time.*

Quality of service (QoS) implementations could be improved if the network could adapt to the context data monitored in the network itself as well as produced by the users. Traditionally the management of communications systems (services, computing components and networks) has been the responsibility of human administrators. In the emerging area of autonomic computing those actions are completely different. Strassner defines in [5]: "Autonomic Computing is a self-government system in which semantic policy management enables business needs to drive the services and resources available" and Kephart defines in [5]: "Autonomic computing are computing systems that manage themselves in accordance with high-level objectives from humans". Over this perspective it seems clear that autonomic systems are emerging as the solution for managing the increasing complexity that is observed in current pervasive and ubiquitous communication systems. In our own words:

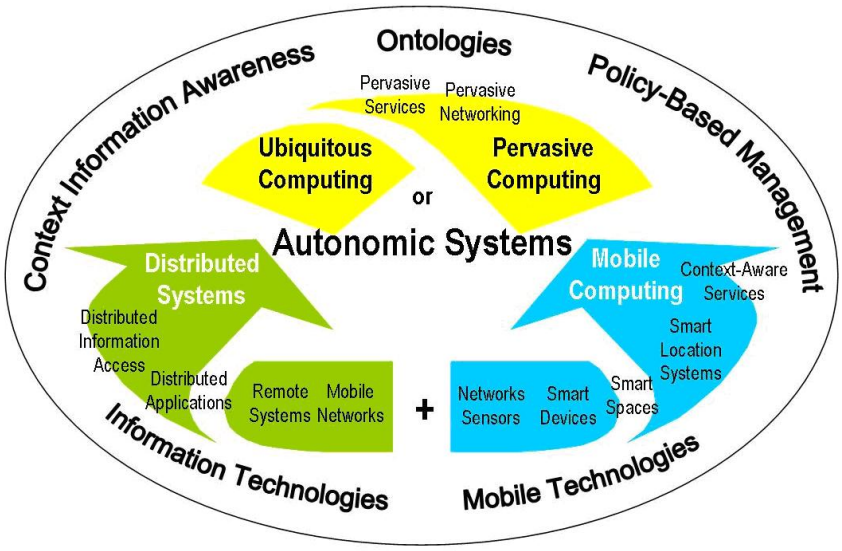


Figure 3-1. Autonomic Systems Composition.

Definition 4

Autonomic systems *are the result of mobile computing technologies and information technologies interacting and cooperating between them for support the creation, authoring, customization, deployment, execution and also management of communication services [6]. The interaction is supported by the high semantic levels in the context information using ontologies and other information technologies such the policy-based paradigm for managing services and networks.* These interactions are shown in Figure 3-1.

Context information is the driving force of any closed loop automated management system that eventually, under the guidance of higher level management rules (policies), and coordinated under a common lexicon that allows the fully interaction between humans and systems leads the autonomic network management.

Definition 5

If we centralize the autonomic computing towards the field of Network Management, then ***Autonomic Management*** *are those technological and software mechanisms dedicated towards increasing the self-managing aspects of the systems and coping with the complexity, heterogeneity, dynamicity and adaptability required in the modern communication services and systems.*

Then to bring autonomic management concept to reality the trend is based on the convergence of advanced pervasive systems with engineering technologies (networking, service, and knowledge also information technologies) and the Internet. Technology advances and the evolution in the communication services have overcome by one side, the idea of everyday devices with embedded technology and connectivity such as computing devices become progressively smaller and more powerful and by other side, the cooperation of diversity systems for supporting multiple services as result of the convergence and interaction of pervasive computing and autonomic computing systems opening the door to the emerging autonomic communications.

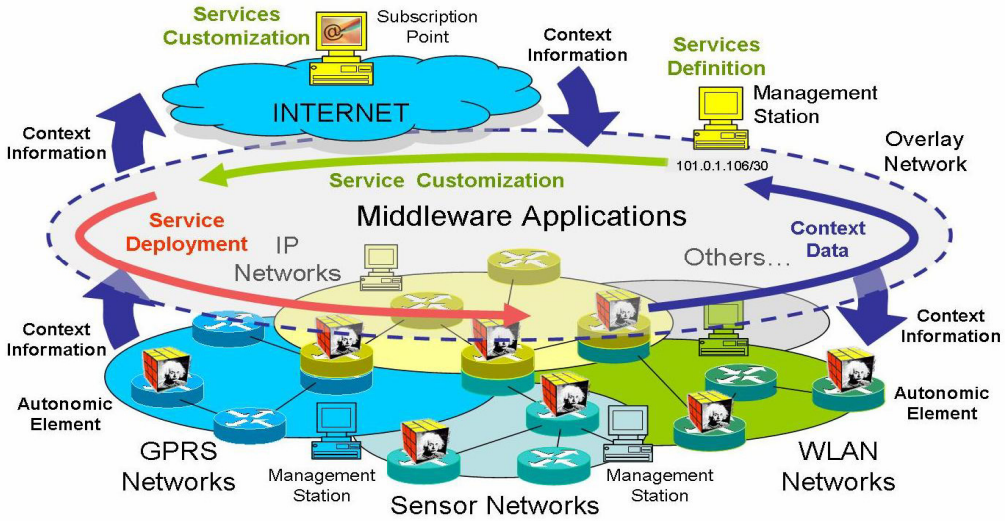


Figure 3-2. Context Information Role in Autonomic Communications Environments.

However nowadays when we talk about heterogeneity of computer systems and technologies there are inherent scenarios with many systems and devices and with different techniques and mechanism for generating, sharing and transferring information amongst each other. In addition to that, different and inter-related management systems and multiple technological platforms supporting services are brought into these scenarios so that the information models that each system use, at different abstraction levels that they have, are independent and specific. Desiring interoperability, the way to achieve it is the efficient and clear interaction between the systems by creating information models that support the free exchange of information for systems management purposes - one of the main objectives of the autonomic computing. Then with the increasing heterogeneity of enormous computer systems, the inclusion of mobile computing devices, and the combination of different networking technologies like WLAN, cellular phone networks, and mobile ad hoc networks the typical management

activity is set as an activity difficult and almost impossible to done. The need for autonomic management is thus obvious.

Based on our extensive literature survey, we have broadly categorized below a set of definitions for autonomic communications. There are many types of context, which participate in the autonomic systems promoting and supporting autonomic communications and their management (see Figure 3-2).

3.2 Context Information for Autonomic Communications

Any kind of activity, including the communication between humans, is surrounded and influenced by context. In the same manner that a hand gesture or a word has different meanings depending on the context or the situation they are expressed, the users of any service are also surrounded by their context when they interact with applications. It must be taken into account that context can be referred to real or physical characteristics, as well as characteristics of the virtual world that circumscribe the performance of the application.

3.2.1 Taxonomy of Context Information

The classification of context information is not an easy task due to its extreme heterogeneity. So this taxonomy could be performed in a multiple ways and perspectives. In this section, we try to make a review of different kinds of classifications [7] (most of them orthogonal and compatible). For instance in a first approximation, context information can be classified by the following characteristics:

- **By its persistence:**
  - Permanent (no updating needed): Context, which does not evolve in time, remaining constant for the length of its existence. (e.g., name, ID card)
  - Temporary (needs updating): Part of the context information that doesn't remain constant. (e.g., position, health, router interface load)
- **By its medium**:
  - Physical (measurable): Referred to the context information that is tangible, e.g., geographical position, network resources, temperature (it is likely that this kind of information will be measured by sensors spread all over the network).
  - Immaterial (non measurable by means of physical magnitudes): The rest of the context information, e.g., name, hobbies (it is likely that this kind of information will be introduced by the user or customer themselves).
- **By its relevance to the service**
  - Necessary: Part of the context information that must be retrieved for a specific service to run properly.
  - Accessory: Additional context information which, although it is no necessary, it could be useful for a better service performance or completeness.
- **By its temporal characteristics**
  - Static: In this category we can put the context that does not change very quickly. For example, the temperature along the day.

- Dynamic: In this category we can find the context that changes quickly, i.e. the position of a person who is driving.

- **By its temporal situation:**
  - Past: This category is for that context which took place in the past. For example, an appointment for yesterday. This category could be assumed as a context history. The context history contains all your previous user contexts. A context trace is a subset of user contexts in the context history. A context trace will only contain the contexts that are related to such situations.
  - Present: This category is for the current context: where am I at this moment, etc.
  - Future: In this category we can find the context that had been scheduled and stored previously for future actions i.e. the venue where a meeting will be held tomorrow morning. The future context can be user contexts that either the systems or the users can predict and describe. For example, activities in your planner. Prediction of future User context would be useful when user moves or at the time of subscribing for a service.

This first classification concerns how context information is gathered and how this information evolves during the time. This classification also gives rise to some issues about how context information should be managed regarding for example its evolving characteristics. Also from this classification we can consider that it would be necessary to place a timestamp and a period of validity associated to each piece of information. Figure 3-3 shows a first approximation in how context information can be classified by the properties described previously.

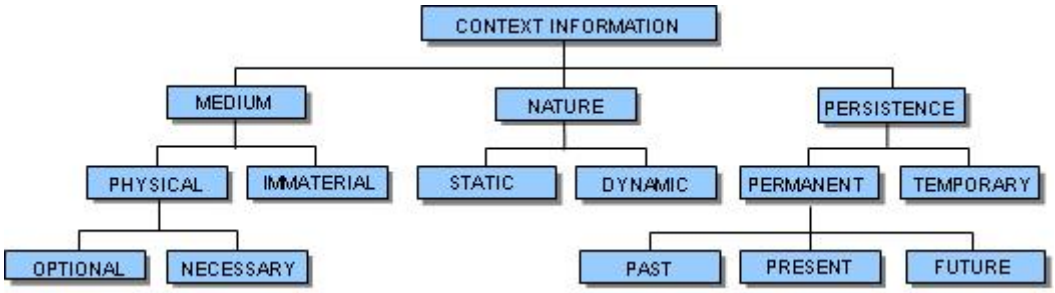


Figure 3-3. Context Information Data Model.

Nevertheless we need more specific types of context information in order to structure and distribute this information throughout the proposed representation model. A classification of context types will help application designers decide the most likely pieces of context that will be useful in their applications. As a starting point five fundamental types of context information could be defined as follow:

- Person information
- Location information
- Environment information
- Task information
- Object/device information

All this information should be mapped on a context model capable of representing all the relationships among different types of contexts. This is the topic of the next section, namely context modelling. We put forward two proposed context models, one based on the entity-relationship model and one based on a graphical, UML-based approach.

3.3 Context Modelling and Storage Representation

3.3.1 Context Modelling

In order to achieve context awareness the information regarded as context has to be modelled in a generic fashion so as for it to be able to be processed by and exchanged amongst distinct entities. The need for context models is denoted by the fact that various sources (i.e. sensors) of information produce a corresponding variety of data that have to be structured and organised under a unifying representation scheme. It is the same underlying driving need that every communication protocol shares, that of establishing a common basis of understanding between different and diverse entities.

The subject of context modelling for wireless environments has not been adequately addressed by the research community in a systematic manner [8]. The problem lies on the fact that every effort towards a context-aware system for wireless networks assumes a different context model to serve its own purposes, constituting thus the solution as non-interoperable with other approaches. To achieve a unified approach towards context aware mobile ad hoc networks in particular there is the obvious need for a generic context information representation through the means of a specific model. This will enable the seamless integration of information in the systems [9][10].

The cornerstone of enabling efficient and constructive context-awareness for autonomic systems is the design of the utilized context model. The simplicity, ease of deployment, performance, scalability, applicability and usefulness of the framework that will tender for context-awareness in autonomic systems is dependent on the context model. Context modelling is thus significant for the system's operation and, as such, inherently linked to its implementation. Based on the analysis of existing context models we set the grounds for our proposed context model in order to cater for the requirements posed by the previously presented taxonomy and provide justification for our design choices.

We put forward and present two different context models, each bearing its own benefits and characteristics and then briefly study related work on the area. While the first model, the one based on the entity-relationship model, is more generic in nature and can cater for a variety of context information, the second model is mostly targeted to tender for context information in resource constrained pervasive environments. This provides the justification for us proposing two different context models that share though the same underlying storage mechanism, namely XML. The two context models are not contradicting each other, rather complementing each other under different applicability scenarios.

3.3.2 Context model based on Entity-Relationship model

We suggest the Entity-Relationship Model Option for modelling context information. Other research work [11][12][13][14] proposes a different point of view when defining context and context information. Basically they define models that refer to the situation that surrounds the user of the service as a physical person. In this user-centric model, definition of context and context model is mainly derived from the fact that the context information is going to be used and stored only on pocket mobile devices used by specific users. We think that this reasoning is not open and generic enough. The user-centric model is limited in this way. On the other hand the information structures regarding each user of the user-centric model should contain for example information of every object surrounding the user (characteristics, utility, and so on), and these same objects could appear in the information structures of other users that are also inside its

scope. So the information about these objects could re-appear inside multiples data structures, generating redundancy and inefficiency in the model. This fact produces lack of scalability. So this redundancy makes this model not suitable to be used over the network or at context information servers.

Modelling context is a complex task, so using a model more abstract and less predefined is more useful to scale its contents in a future, if is needed it. The model denominated entity-centred model can be used to characterize all the context information. Within this model, if the need arises to add new information to the model, it is not necessary to modify the existing entities, rather to create a new entity and establish the convenient relationships with the existing ones. So adding a new entity doesn't require the modification of the content or status of the existing entities (only establish new relationships if it is necessary). The entity-centred model is easily scalable. The motivation and the driving force behind the entity-centred model for modelling context information is the entity-relationship paradigm.

The entity model provides a powerful abstraction of the information needed by context-aware applications. It provides a global scope of the extremely heterogeneous kind of information that could be involved in these applications. In other words, this model could be considered to be a flexible one as it is usually required from context-aware applications. The entity approximation could be seen as a distributed model where the entities contain only their own information and also the type of relationship with other entities, so we can consult the attributes of these other entities if it is needed. This method of representation is a suitable scenario description without reference to any specific actor. So this model can be polled by many different applications regardless which user is using them. The entity model can be thought as a general-purpose way of representing, storing and exchanging context information throughout the network and for this reason we used this concept as premise for the context information model definition.

Context modelling depends on the point of view of the context definition and scope. The model is a first approximation on how to structure, express and organize this kind of information.

- *Classification of Contextual Information*

  Identify the information that could be relevant to context-aware applications and classify it in different classes or categories. This information could be possibly expressed by means of types of relationships or be part of the internal attributes inside the entities.

- *Mapping context information into an entity-relationship model*

  We need to represent all the context information identified using the entity-relationship representation. It is necessary to create a model of predefined entities and relationships suitable to map the necessary information.

  - *definition of generic entities*

    Define the main classes of entities that must exist, making reference to what they represent. (Persons, Objects, Places, Tasks …). Every entity in the final model will be an instance of one of these entity classes.

  - *definition of generic relationships*

Define the classes of relationships that must exist (and their attributes), in order to identify the different possible relationships between entities and what they mean.

- *Representation and implementation tools*
  Identify the possible tools that could be used to represent (*e.g.*, XML) and implement the entity-relationship model and the way to integrate this information model inside the system architecture.

The context model is based on the concepts of entity and relationship. This model is derived from previous definition of context as well as the one given in [15]. The concept of entity can be understood as an object in the context of an object oriented model. The entities can represent anything that could have any kind of influence or relevance at any time in the performance of the activity of an application or service addressed to a user or a managed system.

An entity is composed by a set of intrinsic characteristics or attributes that define the entity itself, plus a set of relationships with other entities. The relationships belong to a specific type, among a set of available types of relationships. The concept of local context of an entity can be understood as the information that characterizes the status of the entity. This status is composed by its attributes and its relationships. Moreover, the relations that can exist between the different entities inside the model, as well as entities, can represent many different types of influence, dependence, association and so on, depending mainly on the type of entities that these relationships connect. Figure 3-4 synthesizes the concept of local context of an entity. It is the sum of all its specific attributes plus the relationships established with other entities inside the model.

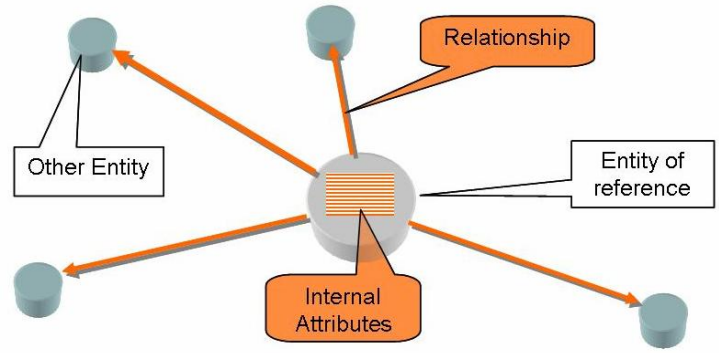


Figure 3-4. Concept of Context Entity

In Figure 3-5, there is a simple high-level example of modelling context by means of the entity-relationship technique. In this example we have taken two possible entities inside a hypothetical scenario as a reference, in this case, an entity that represents a person and an entity that represents a device (a printer device). We represent a possible set of internal attributes of each one and we also represent a few possible examples of relationships between these entities and others that have some kind of influence or connection with the aforementioned (the other entities also have their own internal attributes, but we only represent the ones of the reference entities as an example).

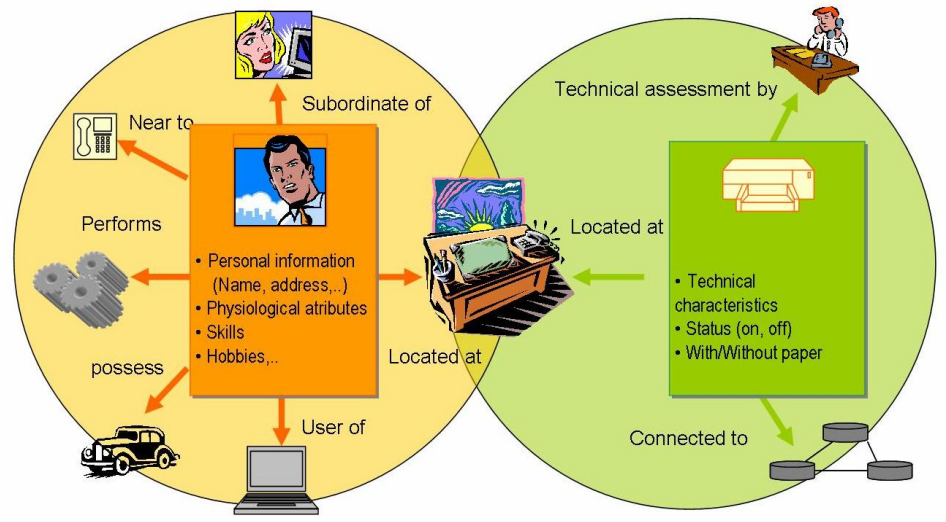


Figure 3-5. High level example.

As presented in the example, there can be many types of relationships depending on the kind of influence or dependency that exists amongst the entities. At a high level these relations could stand for proximity, possession, social, etc. We can consider that the relationships could contain some internal attributes or characteristics as the entities. For example, a relation that express proximity could contain attributes like distance, visibility between entities, etc. With this type of model, we can construct a network of entities and relationships representing the world that surrounds the activity of a context-aware service and that can influence its development.

3.3.3 Context model based on UML principles

The design requirements we have placed on the second proposed context model map those identified in the aforementioned classification of context information and the requirements denoted by the pervasive realm [16][17][18][19][20]. Relevant requirements include *extensibility* to allow for diverse data types to be represented, *limited memory* requirements to store the collected context information, *lightweight processing* so as to cater for the resource-constrained pervasive environment, *interoperability* amongst different context domains and, finally the model should provide support to establish a *degree of accuracy* of the collected context data. The model should tender for *semantics* and be *scalable* so that *expressiveness* can be gained. We chose not to deal with issues regarding *privacy* since we consider that the issue of security is out of scope of this research (security in ubiquitous domains is a vast, standalone research field that could not be adequately coped within a context model). We deem that security should be considered in a different level and not that of context modelling.

Based on these requirements we propose exploiting Unified Modelling Language (UML) design principles for our context model. The main reason for this lies in the fact that context information will be modelled into the system by system engineers. For this reason it is desirable to utilize an easy to understand method of modelling context that will be straightforwardly assimilated. UML is a common, widespread and facile way of modelling systems and this observation leads us to extend its functionality in order to model context for pervasive environments. UML has been used in the past to model context information [21][22], but these efforts have imposed a significant level of complexity on the basic UML options. We differentiate our contribution by using

lightweight mechanisms to expand the fundamentals of UML. Another innovative aspect of our work involves the extended use of semantic metadata to ameliorate the expressiveness of our modelling approach, while at the same time context modelling has been abstracted from the actual data representation allowing for a significant degree of freedom in the design of context-aware operations for autonomic communications. The latter has been a drawback of [21] and [22] where context representation has not been adequately tackled with. This is an equally important aspect as that of context modelling, since it is the actual context representation, or in other words the data model that will be used and incorporated in the overall system.

We assume that every node in the pervasive realm collects its own context information from the sensors available to it. The term sensor here is generic and can, for example, include a battery monitor, a CPU monitor or a GPS receiver. Nodes participating in the pervasive environment can have one or more sensors and every sensor can monitor and report on one or more sources of data.

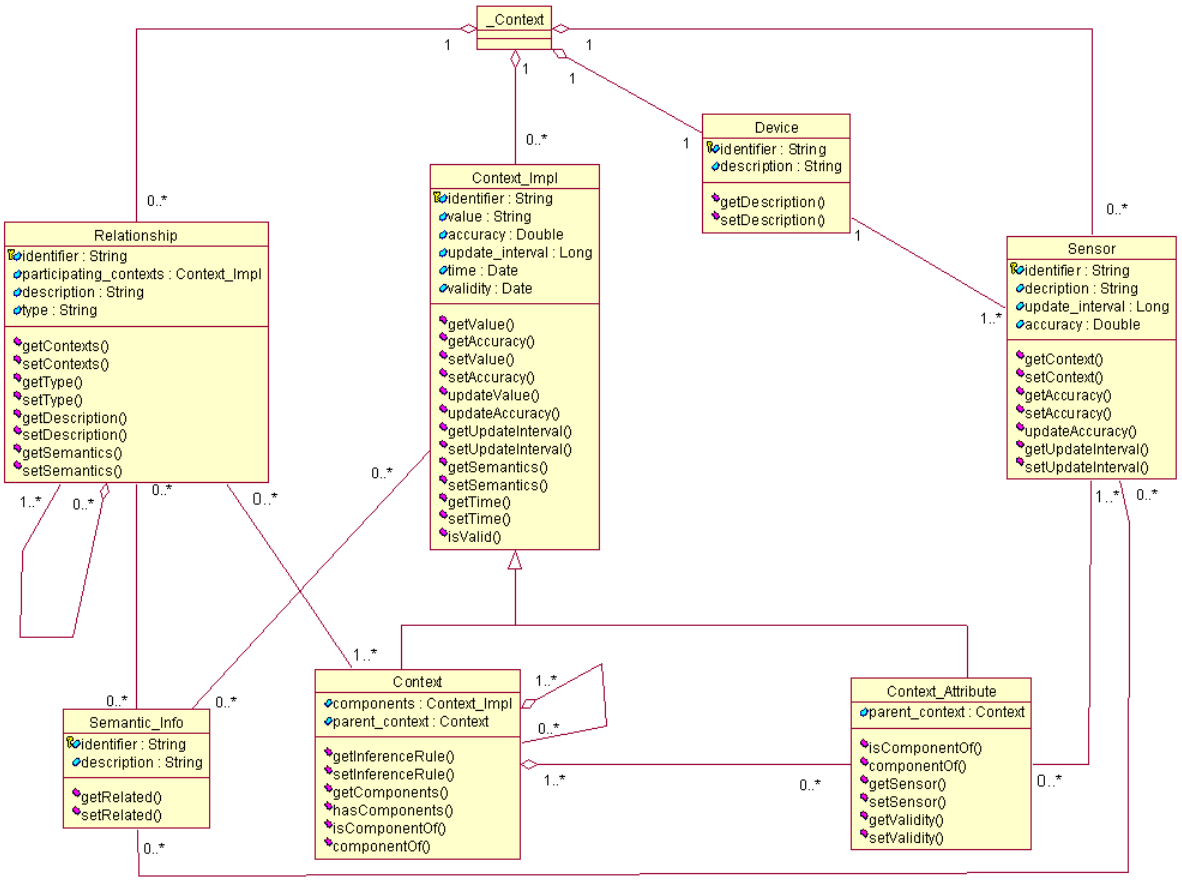


Figure 3-6 Class diagram of proposed context model entities

Our modelling approach is founded on the following general observation: elaborate information is derived from a collection and combination of simpler pieces of information. This can be mapped onto out proposed model under the notion that the broad context of a node consists of higher level contexts that have been deduced from simpler ones. Every context can be partitioned into a number of atomic attributes that have the ability to fully describe the initial context and not be composed of any simpler attributes. In this respect, context is composed of self-explanatory atomic attributes and perhaps other contexts, leading to more complex context structures.

Context composition does not necessarily imply an aggregation of lower-level data to higher-level context information. This assumption, as undertaken by the majority of existing approaches in context modelling would be fairly simplistic and restrictive in terms of expressiveness. One cannot expect elaborate context information, especially in the information rich pervasive environments, to be comprised of purely concatenated or mathematically combined lower-level contexts or simply by using the existing UML relationships such as inheritance and aggregation [21][22]. The existence of semantic-based relationships to infer high level context is undeniably needed. Accordingly, in our proposed context model we exploit relationships. These can span from simple inference rules such as mathematical functions to semantic or user-defined operations, as for example time-constrained functions.

Our proposed model incorporates semantic information regarding context, attributes, sensors and their relationships. For every sensor, context and attribute we store metadata information that describe its functionality, operation or meaning accordingly. This descriptive information is available for both human understanding of the model and its applicability as well as for machine processing with the potential use of ontologies to identify semantic proximity and pattern matching.

Figure 3-6 presents a general UML representation of our proposed context model. More specifically this UML representation depicts the general context of a mobile node and the various components that comprise it together with their interrelationships. Specific contexts can be modelled by extrapolating and using this model, as will be shown at a following stage. The model is expressive and extensible, since custom, user-defined types of context with relationships and semantics can be introduced. Accuracy values for collected contexts are also supported. There are 6 entities that form the basis of our context model as described in the following (we do not provide a reference to the attributes and operations of the classes of the model, since the naming constructs as shown in Figure 3-6 are deemed self-explanatory):

- **Context:** It refers to the overall context that describes a node and its surroundings. It is an abstract, container class of all the contexts that are of use to the mobile node.
- **Device:** This class contains information regarding the device of the node and its configuration. Specifications and operation parameters of the device are its attributes.
- **Context_Impl:** This generic class serves as a super class for the entities of contexts and their context attributes. These two entities share common characteristics and this class serves the purpose of mapping these commonalities.
- **Context:** This class refers to a complex context that can potentially comprise other contexts or context attributes or exist in a standalone fashion. A relationship class is always associated with this class to act as an inference rule from lower-level contexts or attributes. Semantics are also associated to them. Accuracy and validity is modelled based on relevant information from associated relationships and comprising attributes/contexts.
- **Context Attribute:** This class represents the entity of context attributes. These are not associated to relationships since they are derived directly from sensors. Their association is only to parenting contexts and semantic information. Time validity of the information is derived from the “freshness” of the monitored information.
- **Relationship:** This class represents inference relationships for contexts to be deduced from attributes and other contexts. These can be mathematical formulations,

UML relationships, logic rules or user-defined operations. A relationship can be comprised of other relationships and is associated with semantic information.

- **Sensor:** This class refers to the sensors available to the device of the node. The data they monitor are fed to context attribute classes, thus the association between the two classes. Sensor accuracy and update intervals for collecting context are incorporated in this class.
- **Semantic_Info:** It refers to the semantic information associated with every other entity of the model. The structure of the semantics comprised an identifier, a common name and a textual description containing related keywords.

For every type of context information that is of interest to the system's operation, an individual context model is provided, which can be as simple as containing a single attribute, e.g. time that cannot be broken into simpler attributes, or very complex comprising many high-level contexts, attributes and inference relationships. We argue that a generic mechanism to solve the context modelling problem cannot be formulated, in the sense that it is always going to be the administrator who will provide the intelligence to the system to be able to compose complex contexts and infer useful knowledge. The context model we propose is generic to allow the administrator to express any context she wishes incorporating any available semantic information. Automated modelling of higher level contexts cannot be considered since this would require advanced artificial intelligence techniques (that are extremely consuming in resource utilization).

3.3.4 Related Work on Context Models

The context models proposed so far in the literature span a plethora of approaches and concepts, attributing to the aforementioned discussion on custom and not standardized context modelling selection. The most relevant context modelling approaches are presented in this section [23]. The classification is performed from the point of view of the data structures used to exchange contextual information in the respective system. We consider that it is the best and most clear way to understand and contain all the main properties that an information model has to cover.

Key-Value Models

The model of key-value pairs is the most simple data structure and in consequence one of the most used for modelling information. Schilit et al. [24] used key-value pairs to model the context by providing the value of specific context information (e.g. location information) to an application as an environment variable.

Mark-up Scheme Models

The most common characteristic of all mark-up scheme modelling approaches is their hierarchical data structure consisting of mark-up tags with attributes and content previously established. In particular, the content of the mark-up tags is usually recursively defined by other mark-up tags. Figure 3-7 refers to a typical representation example.

```
<xml version="1.0" encoding="UTF-8"?>
<Entity Model xmlns:xsi="XMLSchema-instance"
xsi:noNamespaceSchemaLocation="//Location.xsd">

<Entity>
<Instance Identificator>
</Instance Identificator>
<Attributes>
</Attributes>
<Relationships>
</Relationships>
</Entity>
```

Figure 3-7. Mark-up Scheme Model Representation Example.

Typical representatives of this kind of context modelling approach are the profile-based models that built upon a serialization of a derivative of the Standard Generic Mark-up Language (SGML), the super class of all mark-up languages such as the popular XML. An example of this approach is the Comprehensive Structured Context Profiles (CSCP) [25].

Graphical Models

The highlighting attribute of graphical models is its simplicity to be understood by humans. A set of well defined graphics with functional and relational meaning is the way to represent actors and properties in a system. Figure 3-8 shows a graphical model representation example. An extensive and very well known general purpose modelling instrument is the Unified Modelling Language (UML) which has a strong graphical component (UML diagrams). Due to its generic structure, UML is also appropriate to model context information.

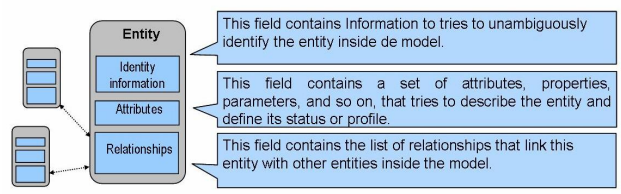


Figure 3-8. Graphical Model Representation Example.

An example is the graphics oriented context model is introduced by Henricksen et al [21], which is a context extension to the Object-Role Modelling (ORM) approach [22]. In ORM, the basic modelling concept is the *fact* and the model of a domain using ORM involves identifying appropriate *fact types* and the roles of the *entity types*.

Object-Oriented Models

Object oriented context modelling approaches try to get the advantages derived from the use of the object-oriented paradigm to cope with the problems arising from the dynamics of context. The details of context processing are encapsulated at object level and hence hidden to other components and the access to contextual information is provided through specific interfaces. Figure 3-9 shows a simple Object-Oriented representation example.

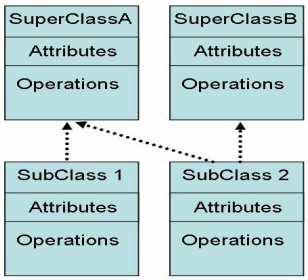


Figure 3-9. Object-Oriented Model Representation Example.

A representative for this kind of models is the *cues* approach developed within the TEA project [26]. The concept of *cues* provides an abstraction from physical and logical sensors. A *cue* is regarded as a function taking, as input, the value of a single physical or logical sensor up to a certain time and providing a symbolic or sub-symbolic as output. Another approach is the Active Object Model of the GUIDE project [27] Again, the chosen approach has been primarily driven by the requirement of being able to manage a great variety of personal and environmental contextual information while retaining scalability.

Logic-Based Models

A logic system establishes conditions that derive in concluding expressions and facts related with other(s) condition(s) (the reasoning or inference process). These conditions are formally described as a set of rules. In a logic based context model, the context is consequently defined as facts, expressions and rules and usually the contextual information is *added to*, *updated in* and *deleted from* a logic-based system. Figure 3-10 shows a set of conditions as logic-based model representation example. One of the first logic based context modelling approaches was presented by McCarthy [28].

$$S1 = [\ U|u| = \{\ Birds\ \}\ ]$$

$$S2 = [\ U|u| = \{\ Flies\ \}\ ]$$

$$S3 = [\ U|u| = \{\ Pinguins\ \}]$$

$$B = [\{\ Presence\ air\} \text{ and } \{wings\} \text{ and } \ldots]$$

$$S3 = S1 \rightarrow S2 \mid B$$

Figure 3-10. Logic-Based Model Representation Example.

Ontology-Based Models

Ontology is an instrument to specify concepts and interrelations. Ontology-based models are particularly suitable for information concerning our daily life that needs to be interpreted by computers. Under this assumption it is not difficult to imagine how powerful such a tool an be in context-aware applications and the media systems world in general. One of the first attempts for modelling context with ontologies is the job proposed by Ötztürk and Aamodt [29].

A promising emerging context modelling approach based on ontologies is the CoBrA system [30]. This tool provides a set of ontological concepts to characterize entities such as persons, places or several other kinds of objects within their contexts. The CoBrA system uses broker-centric agent architecture to provide runtime support for context-aware systems, particularly in Intelligent Meeting Rooms, a prevalent scenario of ubiquitous computing environments. In addition, COBRA-ONT [31] is a collection of

ontologies in the CoBrA architecture for smart spaces (e.g. intelligent meeting rooms, smart homes, and smart vehicles).

3.3.5 Storage Mechanisms

Modelling context information is one side of the twofold issue of providing context awareness in autonomic management system. The other side refers to the actual representation of this data, so as for the system to be able to process and handle them. We propose the XML Language to represent the data modelled through the proposed context model we put forward in previous section.

Using this language has several advantages:

- XML is a mark-up language for documents containing structured information.
- The use of DTDs (Document Type Definition) and XML Schemas in order to validate the documents created automatically when representing the context information.
- XML is commonly used as a mechanism to exchange and store data since it is extremely interoperable.
- The use of XQuery, as a powerful search engine, to find specific context information inside the XML documents that contain all the information related to a specific entity. These queries can select whole documents or sub-trees that match certain conditions

We should also state that the use of XML has some drawbacks:

- XML is a hierarchical language that restricts the architecture that we proposed, which is database oriented.
- The use of XML increases the amount of information that the context model needs to exchange.

The major requirement is to minimize the amount of context information transferred throughout the network so as to overcome potential problems. The inherent benefits that autonomic management obtains from context awareness can be hindered, if an efficient storage mechanism is not selected. We experimented with compression techniques that are specifically targeted for XML documents, since the tagging and individual characteristics of XML can be exploited to reduce the size of generated documents. The results of our custom benchmarking tool are encouraging; we opted though against using such techniques since they require considerable processing time and subsequently have a detrimental effect on node batter power. We propose instead to exploit common characteristics of context in order to achieve a degree of optimization, such as:

- Context aggregation: Context information is periodically aggregated and average values sampled over time are actually transmitted, not every single change in monitored context.
- Normalization of context values: When it is possible and without loss of precision, context values are normalized in certain ranges, allowing for smaller data to be transmitted.
- Threshold criteria: Criteria associated with specific contexts may result in context transmission only when certain thresholds regarding context changes have been exceeded.

3.4 Discussion

In this chapter, we presented an overview on descriptions regarding the context-related research activity for this deliverable; we have identified major problems and their possible solutions, as well as basic concepts pertinent to the context information in autonomic communications systems as likewise the required functionality and interactions for provisioning such services. We conclude by providing pointers for open issues and general conclusions.

3.4.1 Open issues and challenges

We hope to investigate the feasibility of adding value to the range of customer services by including context data at network-level granularity. This would require adapting additional parameters of user- and network-centric information when provisioning for end-to-end services. The goal is to achieve better personalization of user experience as per service type. To meet these challenges there is a need to adopt an enabling technology that would facilitate rapid and efficient prototyping of services. This is an added challenge that would significantly add value to the autonomic communications provisioning research community.

3.4.2 Conclusions

Although there have been many context-aware systems and applications tested over the last decade, most of them are still prototypes and only available in the research labs and the academia. One of the main drawbacks lies in the complexity of capturing, representing and processing the contextual data. We feel that the pervasiveness of context-aware system will only be appreciated if the context can be interpreted properly.

In personal communications people are using implicit information to increase the amount of delivered information. This implicit information is related to the context affecting the individuals or the subject they are dealing with. This is a feature that makes people to react appropriately at concepts involved in a conversation, which is not said explicitly by any of the interlocutors. Currently computers are not able to take and process all advantages that the context information has in humans' dialogues.

Clearly, it would be desirable to allow applications hosted in computers to interact with human beings making use of at least part of the information laying in the surrounding context. Due to the nature of the communication process the first step is that users and devices make explicit all the information relevant to a given situation. Nevertheless this is not easy; the problems start when most users do not know which information is potentially relevant and, what information to announce or broadcast.

We also recognized the difficulty of distributing acquired context information broadly among massive context-aware applications across different networks.

4 Automated QoS Management for Fixed IP Networks

With the prospect of becoming the all-service network of the future, the Internet needs to evolve to provide services such as e-commerce, real-time voice and video applications, multimedia services, and so on. Such services require a minimum guaranteed Quality of Service (QoS). However, the current Internet environment provides only the Best Effort (BE) service which does not guarantee delivery of data and treats all packets equally. When the volume of network traffic grows, congestion at nodes may occur and may slow down the delivery of packets or if the congestion continues and becomes severe, packets are dropped.

In order for QoS to be provided in a practical manner, QoS oriented service models, such as Differentiated Services have been proposed [32]. In such a QoS oriented service model, when an Internet user/customer wants to use QoS to support some service, they have to negotiate a Service Level Agreement (SLA) with the network provider to request the resources and the associated QoS guarantees before they can actually use the service.

This negotiation process, however, in most cases is not automated. Today the service level agreement (SLA) is usually agreed, either verbally or in writing, by both the user and the provider when a user signs up for a service. The provider stores it in some repository and uses the technical part of the SLA -referred to as the Service Level Specification (SLS) - to condition the traffic sent from the user. To change the SLA, normally a user has to contact and negotiate with the authority of the network provider, which then manually changes it.

Compared to manual service negotiation methods, automated service negotiation offers higher degrees of flexibility to customers and the network provider by reducing their time to request or to gain access to services. To enable this flexibility, however, the network provider must also deploy automated mechanisms to adapt to the changing conditions and QoS requirements that are introduced by an automated service negotiation process. These mechanisms need to ensure that the provider's resources are adequate to support the QoS features/parameters requested by the automated service negotiation process and also that the QoS monitoring functionalities of the network provider are always up-to-date and configured to measure the relevant QoS features/parameters, associated with the services introduced by the service negotiation process.

Towards this direction of automated QoS management, in the rest of this section we will present an architecture for automated service negotiation and service provisioning for achieving QoS in fixed IP networks.

4.1 Framework for automated QoS management

The functional blocks of the proposed architecture for automated QoS management are depicted in Figure 4-1.

In the remaining of this section we will briefly describe the basic functionality of the blocks in the architecture and we will focus on the interfaces between:

a) the SLS Ordering and SLS Order Handling blocks, and

b) the SLS Order Handling and QoS Monitoring block

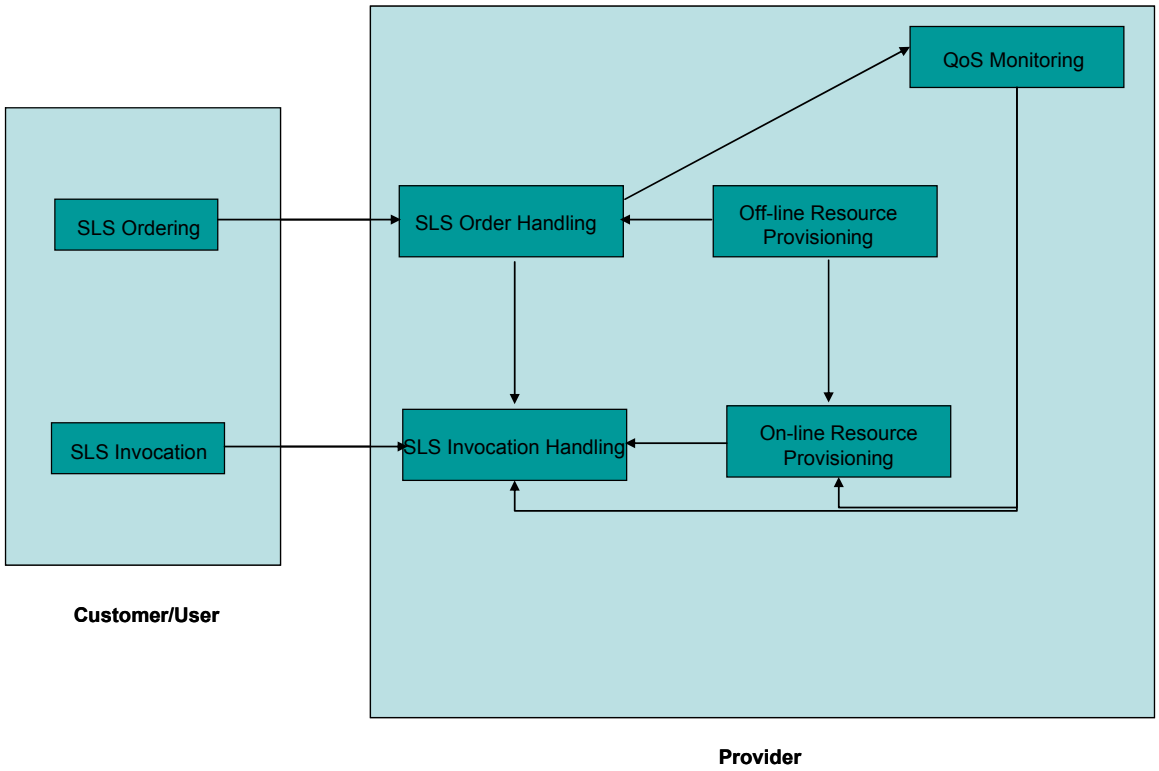


Figure 4-1: Architecture for automated QoS management.

- **SLS Ordering**

SLS Ordering is the functional block implementing the customer/user side of the SLS negotiation process. Its role is to negotiate services with the provider and if the negotiation process is successful, to subscribe to these services.

- **SLS Order Handling**

SLS Order Handling is the functional block implementing the provider side of the SLS negotiation process. This block maps incoming SLS requests to the provider's resources and decides whether the resources are adequate to accommodate the SLSs in question.

- **Off-line Resource Provisioning**

Off-line Resource Provisioning is the functional block that in an off-line manner tries to optimize the resources in a provider's network based on some optimization objectives. This block provides the SLS Order Handling block with the resource availability matrix (RAM), which is then used by the SLS Order Handling during the SLS negotiation process to determine whether incoming SLS requests can be accommodated.

- **SLS Invocation**

SLS Invocation is the functional block at the customer/user side that is responsible for informing the provider that the customer/user want to invoke (activate) an already subscribed SLS and to inject traffic to the provider's network.

- **SLS Invocation Handling**

SLS Invocation Handling is the functional block at the provider side that deals with requests for invoking SLSs and based on the current 'run-time' status and configuration of the resources it decides whether the SLS can be invoked or denied/postponed until

the status and configuration of the resources becomes such that it can meet the QoS requirements of the SLS in question.

- **On-line Resource Provisioning**

On-line Resource provisioning is the functional block that tries to optimize on the fly the resources in a provider's network. Given the current network configuration, as provided by the Off-line Resource Provisioning block, this block tries to re-optimize/reconfigure the use of the resources at certain local points (e.g. at congested nodes) without invoking a network-wide reconfiguration. This block provides the SLS Invocation Handling block with the up-to-date implemented network configuration.

- **QoS Monitoring**

QoS Monitoring is the functional block that in real-time monitors the state of the resources. The monitored information is then fed using some technology such as Web Services to the On-line Resource Provisioning block to drive the on-line reconfiguration of resources. The outcome of QoS monitoring is also used to assist the SLS Invocation Handling block in deriving the decision on whether an SLS can be invoked or has to be denied/postponed.

As aforementioned, the SLS negotiation process (that is the relationship/interface between the SLS Ordering and SLS Order Handling block) is performed in most cases manually. Automation of this process will provide greater flexibility but requires the existence of a protocol/mechanism to implement the automation of this process. The proposed protocol for implementing this automation is called Service Negotiation protocol (SrNP), developed originally in the context of the TEQUILA [33] project. Additionally, the configuration of the QoS Monitoring block is also usually done manually by the provider. Given the automation of the SLS negotiation process, it is clear that the capability to automate the process of configuring the QoS Monitoring block so that the parameters that need to be measured are dynamically and without manual intervention configured to follow the QoS parameters in the subscribed SLSs, is imperative for the proper operation of the architecture. The proposed mechanism for implementing the automated provisioning of the QoS Monitoring block based on the specified QoS parameters (that is the interface between the SLS Order Handling and QoS Monitoring block) is QoSSL (Quality of Service Specification Language) [45].

In the next sections we will elaborate on the functionality of SrNP and QoSSL as parts of this broader architecture for automated QoS management. For more details regarding the functionality of the blocks the interested user can refer to the relevant deliverable of the TEQUILA project [33].

4.2 Service Negotiation Protocol (SrNP)

4.2.1 Related Work in Service Negotiation Protocols

For a comparative study of protocols for dynamic service negotiation the interested user can refer to [34]. The main benefit of SrNP against other protocols is in terms of SLS transparency. This is because SrNP does not assume any specific format for the SLS. Other protocols adopt specific formats for the SLS that is being negotiated in spite of the fact that there exists no universally accepted format for SLS specification yet.

4.2.2 Negotiation Protocol Requirements

SrNP design is driven by the following requirements.

Functional requirements

- The negotiation protocol should provide for primitives to enable the process of negotiations between two or more parties (negotiating parties). Generally speaking, the negotiation process is a process where several parties are seeking for an agreement on a number of commonly understood issues e.g. on an SLA.
- There should be a clear distinction between the primitives offered by the negotiation protocol and the negotiation logic. The term 'negotiation logic' denotes the logic according to which negotiations are conducted on behalf of each negotiating party. Negotiation logic should be specific to each negotiation party, being subject to its business policies and operational capabilities, and as such, is considered application- and domain-specific. In essence, the negotiation protocol should provide a service to the negotiation logic i.e. it should be seen as a layer based on which application-specific negotiation logic could be built.
- The negotiation protocol should not duplicate but complement the functionality of existing, widely deployed, standardised protocols. For SLA agreements (QoS negotiation), the corresponding negotiation protocols should not duplicate (aspects of) the functionality of existing QoS-signalling or reservation or session control protocols (e.g. RSVP).
- The targets of the negotiation logic for the particular negotiation process may not be static; rather they may change depending on factors outside the particular negotiation process. It is therefore mandatory that the negotiation protocol allows for the negotiations to continue further beyond the first candidate agreement found satisfactory by all parties, as long as there are parties with related pending issues to resolve.
- The negotiation protocol should lead at convergent negotiation processes. Appropriate mechanisms should be provided at protocol layer, for ensuring that the negotiation process can terminate successfully or unsuccessfully in finite steps and in a reasonable time period, as deemed necessary by (the negotiation logic of) each of the negotiating parties.
- The negotiation protocol should be independent of the underlying transport and network protocols. In fact, it should be able to operate with multiple such protocols.

Non-functional requirements

- The negotiation protocol should provide for secure and reliable communication.
- The negotiation protocol should be expandable in terms of additional negotiation primitives.
- The negotiation protocol should be able to support a number of simultaneous active negotiation processes.

It is clear, that the above requirements contribute to the openness and therefore the applicability of negotiation protocols; they are not specific to SLA negotiations and they could apply to any negotiation protocol.

4.2.3 Negotiation Model

The following assumptions underline the negotiation model to which SrNP has been designed to apply.

It is assumed that the negotiation process involves *two parties* only; one acting in a server role, called the *server*, and the other acting in a client role, called the *client*. The roles are exclusive to the parties that is, a party cannot act in both roles in the context of a particular negotiation process. Following the usual distinction between client and server roles (client requests, server responds), in the context of a negotiation process, these roles are distinguished in that agreements can only be pursued by the client towards the server. This distinction is in line with the semantics underlying a customer-provider relationship between two interacting parties. Note though that the provider-customer case is also considered valid. For instance, this case may arise in situations where the provider deems necessary to renegotiate SLAs with some customers for improving or lowering the quality of the subscribed services.

It is assumed that the issues under negotiation can be described in a form of a *document*. The target of the negotiation process is then for the negotiating parties to come to an *agreement regarding the content of* (information included in) the document.

Furthermore, it is assumed that all negotiating parties have a common understanding of the semantics and syntax of the information included in the document as well as means for constructing, extracting and manipulating the information in the document. In line with the requirements presented in the previous section, document/information format, construction and manipulation are not of concern to the protocol, but rather of the negotiation logic. Evidently then, SrNP is not specific to any SLA format or to the content of SLAs. It is general enough to apply to negotiating any issues, provided that these issues can be appropriately described in the form of a commonly understood document.

Finally, it is assumed that authentication and authorisation with respect to negotiation aspects are not of concern to SrNP; they are of concern to the negotiation logic that SrNP services. It is also assumed that SrNP uses the services of a reliable and secure transport protocol.

4.2.4 SrNP Overview

SrNP is an *application-layer*, *session-oriented* protocol allowing for sessions to:

- establish an agreement,
- modify an established agreement, and to
- delete an established agreement

SrNP sessions are initiated by the client.

Agreement Establishment Session

Generally speaking, the negotiation process for establishing an agreement is an iterative process, whereby the negotiating parties exchange their views/requirements on the issues under negotiation until an agreement is reached.

SrNP follows a *client-server*, *multi-dialogue-based* approach for realising the necessary interactions between the negotiating parties towards establishing an agreement. Specifically:

First, the client *connects* to the server to initiate a session for negotiating the establishment of an agreement. Within a session, the client initiates *options*. An option is an independent negotiation track allowing for pursuing a particular variation of the

issues under negotiation. Agreement will be eventually established on only one of the open options.

Within each option the client issues *proposals* and the server responds by either *accepting* the proposals or by issuing *revisions*. Proposals and revisions convey the client's and the server's views/requirements on the issues under negotiation, respectively. Through revisions, the server is enabled to respond to the client views/requirements not in a monolithic 'agree/do not agree' manner but, in a flexible, in the spirit of 'I could agree provided that/even if', manner indicating the points of argumentation and suggesting possible alternatives. It is up to the negotiation logic of the client to determine whether to adhere or not to adhere to the suggested revisions in subsequent proposals.

Accepted proposals and concrete revisions can be *cooled* by the client as an indication of provisional agreement; the option then is considered closed unless either party decides to drop it. The option can be *dropped* by the client at any time, while be the server it can be dropped only as a response to a request. However, a negotiation session may be *quitted* at any time by either party.

Eventually the client calls for an *agreement* on either a previously provisionally agreed proposal or on a new one. If the server *accepts* it the negotiation process concludes successfully, otherwise the client may call for another *agreement* until either the agreement is reached or either party decides to abort. When accepting the call for agreement the server also includes the received proposal as a form of 'hand-shaking'.

The client and the server interact in two different levels, in the session and in the option level. Messages in the session level affect the status and the behaviour of the protocol in the option level.

Within an option the client and the server interact in a dialogue (half-duplex) manner; once a party sends information to the other party, the party is blocked until a *valid response* from the other party is received. Specifically, once the client sends a proposal, it is blocked until it receives a revision, an acceptance or a rejection from the server. Similarly, once the server replies to the last proposal, it is blocked until it receives an alternative proposal or a signal of provisional agreement or a rejection from the client. Once the client issues a rejection, the option is dropped.

At the session level the client may at any time initiate options, initiating thus another parallel dialogue (multi-dialogue nature). A request for agreement occurs at the session level and results in pausing open options. In case of failure to reach the agreement, negotiation over the open options may resume as normal. Either party may quit session at any time; the protocol terminates the negotiation process from this party, without waiting any further response from the other party.

To ensure graceful operation, SrNP does not allow a party to be blocked forever, waiting to receive a valid response from the other party. To this end, when a party sends information to the other party within an option, SrNP requires that the party must specify a maximum tolerable time period willing to wait for the other party to respond back. SrNP will reject the negotiation process on behalf of the sending party, if during the specified maximum tolerable time period no valid response from the other party is received. In addition to avoiding communication blocking, this mechanism of SrNP has the intuitive counterpart of 'sent information is only valid for a specific time period'. In addition, the negotiation session is governed by a maximum idle time allowed between any actions in the options level.

SrNP also offers the negotiation features of 'last word', 'need more time' and 'need clarifications'. Specifically:

SrNP provides the means to either party for forcing the negotiations to conclude allowing for a last round. The last round can be invoked by the client by sending a last proposal. If the server replies with a definite agree or reject answer to the received proposal the negotiations conclude. The server is also entitled to reply with a last revision and the negotiations will conclude with the definite answer of the client to the last revision. If the server invokes the last round the client must reply with a definite answer, either to call for agreement on its last proposal or to abort negotiations altogether. In addition to its intuitive counter part ('last word'), this feature offers a lever for enforcing the termination of a negotiation process in finite steps.

SrNP allows either party to request more time beyond the maximum tolerable time period for providing its response to the last received position within a negotiation track. The other party may reply by accepting to grant more time, not necessarily equal to the requested time extension, or by rejecting to extend the tolerable period. This can be useful for example when the server sees that an agreement is likely to be reached shortly after the elapse of the time period specified by the client, or when the client waits for an answer to other correlative negotiation processes in order to evaluate its margin to consent to the server position, or in the case of human interaction for negotiation decision making, etc.

Finally, SrNP provides means to ask for clarifications at the negotiation logic level. In case a proposal or a revision is not concrete with the criteria of the server's/client's negotiation logic, i.e. when important negotiation issues are not sufficiently specified, then clarifications may be requested. The other party then should reply with an appropriately modified proposal/revision.

Agreement Modification Session

During this session, SrNP operates similarly to the agreement establishment session outlined above. In this case, the first proposal to be sent by the client denotes the agreement modifications that the client wishes to make.

Agreement Deletion Session

Once the client has successfully initiated a session for deleting an already established agreement, SrNP allows for the server to respond by either accepting or rejecting the agreement deletion request. The decision for accepting or rejecting the deletion request is taken by the server negotiation logic.

4.2.5 SrNP Messages and Interface

The SrNP messages reflect the negotiation primitives offered by the protocol to the negotiation logic. SrNP messages are distinguished into *client* or *server session* or *option* level messages.

Following the multi-dialogue nature of SrNP, during a negotiation session initiated by the client for establishing/modifying/deleting an agreement, client and server messages are exchanged alternately within the context of an option (one after the other); server messages are sent in response to client messages and vice versa.

Furthermore, SrNP dictates that messages must be exchanged in a particular order, reflecting the natural evolution of a negotiation process. That is, given a message sent by a party, the other party can respond only with specific messages, which SrNP regards as *valid responses* to the message sent. Subsequently, the party, which has sent a message and received a valid response by the other party, can only send specific messages corresponding to the valid responses of the response-message received, and so on until the negotiations are terminated.

The SrNP messages are described in Table 4-1 and their parameters in Table 4-2.

	Message	Description	Valid Responses
SrNP Client Messages			
session	SessionInit	It requests the initiation of a session for negotiating the establishment, modification or deletion of an agreement. It is the first message that the client must send.	SessionAccepted, Quit
	BindProposal	A proposal that signals a call for agreement to conclude the negotiations.	AgreeProposal, RejectBindProposal, Quit
	LastProposal	A proposal that forces the server to either consent to the position of the client carried by the message, quit, or state one last alternative position. LastProposal signals the start of the last negotiation round.	AgreeProposal, LastRevision, Quit
	Quit	It indicates that the client is not satisfied by the responses of the server and does not wish to pursue a better deal any further and, as such, is not willing to continue the negotiations. When reliably delivered to the server, the protocol terminates at both ends concluding unsuccessfully the negotiations.	
option	Proposal	It carries the client's requirements/views on the issues under negotiation, described in a document, which must be commonly understood by the negotiating parties. When used with a new option identifier it initiates a new option. This message is exchanged during negotiations for establishing or modifying an agreement. The carried document is constructed by the client's negotiation logic and the semantics and syntax of the information included in it are transparent to the protocol. This -or the LastProposal message- is the first message that the client must send after the negotiation session has been established.	Revision, LastRevision, Accept, Reject, Clarify, Time, Quit
	Cool	It indicates that the client wishes to establish a provisional agreement on the last server position - either an accepted client proposal or a concrete server revision. A provisional agreement notifies the server that its last position is a satisfactory candidate to finally agree upon sometime later, depending on the availability of other options in this or other correlated sessions with this or with other servers. With this message the option is considered closed; it can only be dropped allowing for no other position exchanging over it.	Reject, Quit

	ForgetIt	It indicates that the option is dropped by the client, either as a result of non satisfactory responses from the server or after a prior provisional agreement because the client now is offered other better options.	Quit
	Time	It requests an extension of the time allotted to the client for responding in the last message from the server.	TimeGranted, Reject, Quit
	TimeGranted	It signals that a previously requested time extension is granted to the server for responding in the last client proposal.	Revision, LastRevision, Accept, Reject, Quit
	Clarify	It requests clarifications at the negotiation logic level on the lastly received document.	Revision, LastRevision, Reject, Quit
SrNP Server Messages			
session	SessionAccepted	It confirms the client's request to initiate a negotiation session for agreement establishment, modification or deletion (cf. SessionInit message).	Proposal, BindProposal, LastProposal, Quit
	AgreeProposal	It indicates that the server consents to the last received proposal on the issues under negotiations (cf. BindProposal/LastProposal messages) and an agreement is reached. The message should carry the last received document by the client as a form of 'hand-shaking'.	
	RejectBindProposal	It indicates that the server rejects the call for agreement received by the client. However, further negotiations may proceed as normal.	Proposal, BindProposal, LastProposal, Quit
	LastRevision	A revision that forces the client to either call for agreement or quit negotiations. LastRevision signals the start of the last negotiation round.	BindProposal, Quit
	Quit	It indicates that the server does not wish to continue negotiations any further. When reliably delivered to the client, the protocol terminates at both ends concluding unsuccessfully the negotiations.	
option	Revision	It carries the server's counter-requirements/views on the issues under negotiation, should the server cannot accept (some of) the respective client's requirements/views as last received (cf. Proposal message). Server's counter- requirements/views are described in a document, constructed by the server's negotiation logic, which must be commonly understood by the negotiating parties. The semantics and syntax of the information included in the document are transparent to the protocol.	Proposal, BindProposal, LastProposal, Cool, ForgetIt, Time, Clarify, Quit
	Accept	It indicates that the server accepts the last received client's requirements/views on the issues under negotiations (cf. Proposal/LastProposal messages).	Proposal, BindProposal, LastProposal, Cool, ForgetIt, Time, Quit

	Reject	It indicates that the server rejects the last received client's requirements/views on the issues under negotiations (cf. Proposal message) entirely without presenting counter-requirements/views.	Quit
	Time	It requests an extension of the time allotted to the server for responding in the last message from the client.	TimeGranted, ForgetIt, Quit
	TimeGranted	It signals that a previously requested time extension is granted to the client for responding in the last server position.	Proposal, BindProposal, LastProposal, Cool, ForgetIt, Quit
	Clarify	It requests clarifications at the negotiation logic level on the lastly received document.	Proposal, ForgetIt, Quit

Table 4-1: SrNP protocol messages.

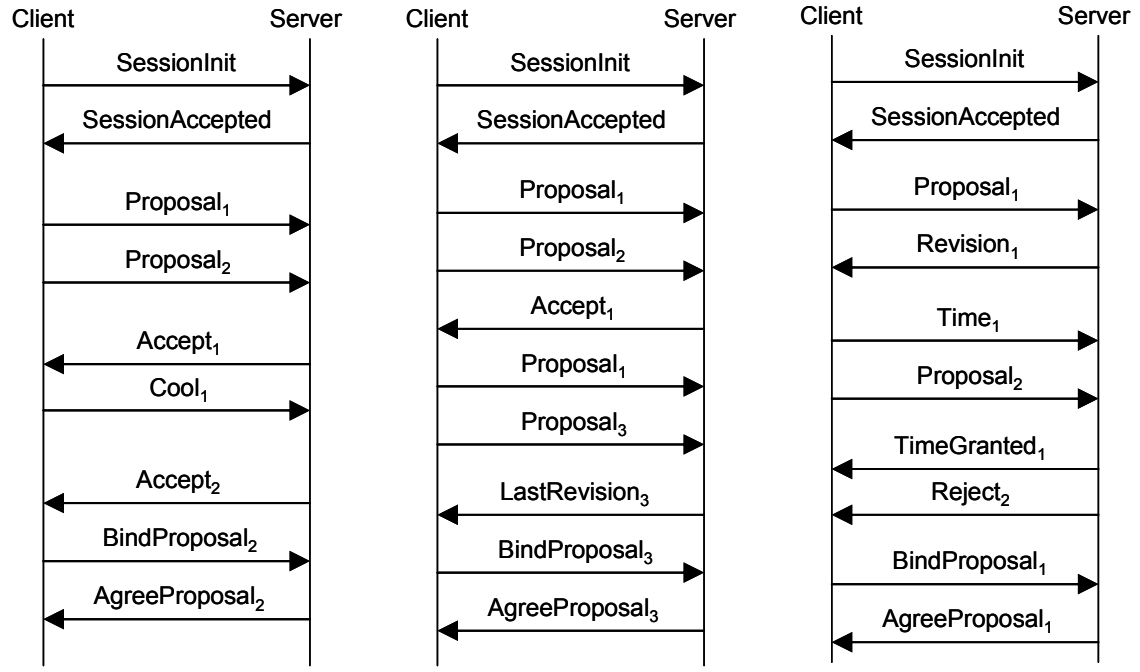
SrNP Header Parameters – common to all messages				
SessionId	Unique identifier of the negotiation session. It is a compound identifier composed by a unique identifier assigned by the client SrNP engine and a unique identifier assigned by the server SrNP engine.			
MessageId	Unique identifier of the sent messages in the locality of a party. It is set by the SrNP engine.			
InResponseTo	The MessageId of the message to which this message is sent as a response. By examining this parameter a party is able to determine whether a received valid response is correct i.e. it truly corresponds to the last message sent by this party. It is set by the SrNP engine.			
TimeToRespond	The maximum time period, in seconds, that a party sending a message is willing to wait for the other party to respond. If this period elapses, the protocol terminates. It is specified by the negotiation logic.			
OptionId	Unique identifier of the option to which the message refers to. It may not be present to some session level messages. It is specified by the SrNP engine.			
Message	**TimeToRespond**	**OptionId**	**Parameter**	**Description**
SrNP Client Message Parameters				
SessionInit	√		SessionType	The type of the negotiation session the client wishes to initiate. It takes three possible values: newAgreement, modifyAgreement, deleteAgreement.
			ConctactAddr	The client contact address where the server will send subsequent SrNP replies.
			MaxIdleTime	The maximum idle time, in seconds, allowed between any actions from the client in the options level. If this period elapses, the protocol terminates.
			AgreementId	In cases of sessions for modification/deletion of established agreements, it is the unique identifier of the agreement to be modified or deleted as set by the server (cf. AgreedProposal message). In cases of sessions for establishing a new agreement, it is left unspecified.

BindProposal	√	√	Document	The document describing the client's requirements/views on the issues under negotiations. It is constructed by the client's negotiation logic.
LastProposal	√	√	Document	As in BindProposal.
Quit			Reason	The reason for terminating the negotiation process. It is specified by the client's negotiation logic.
Proposal	√	√	Document	As in BindProposal.
Cool	√	√		
ForgetIt		√	Reason	The reason for dropping the option. It is specified by the client's negotiation logic.
Time	√	√	TimeRequest	The additional time requested by the client for responding in the last message received by the server.
TimeGranted	√	√	TimeGranted	The additional time granted by the client to the last time extension request sent by the server.
Clarify	√	√	Document	The document requesting clarifications on the previously received server's requirements/views.
SrNP Server Message Parameters				
SessionAccepted	√			
AgreeProposal		√	Document	The last document received with a call for agreement describing the client's requirements/views on the issues under negotiation (carried by a BindProposal/LastProposal message).
			AgreementId	A unique identifier of an agreement It is specified by the server's negotiation logic.
RejectBindProposal	√	√	Reason	The reason for rejecting the last call for agreement issued by the client. It is specified by the server's negotiation logic.
LastRevision	√	√	Document	It describes the server's counter-requirements/views on the issues under negotiation based on the respective client's requirements/views, as last received by a Proposal/LastProposal message.
Quit			Reason	The reason for terminating the negotiation process. It is specified by the server's negotiation logic.
Revision	√	√	Document	As in LastRevision.
Accept	√	√		
Reject		√	Reason	The reason for dropping the option. It is specified by the server's negotiation logic.
Time	√	√	TimeRequest	The additional time requested by the server for responding in the last message received by the client.
TimeGranted	√	√	TimeGranted	The additional time granted by the server to the last time extension request sent by the client.
Clarify	√	√	Document	The document requesting clarifications on the previously received client's requirements/views.

Table 4-2: Parameters of the SrNP protocol messages.

4.2.6 Message Sequence Charts (MSCs)

Figure 4-2 and Figure 4-3 depict the exchange of SrNP protocol messages in a number of typical negotiation processes initiated for agreement establishment or modification.

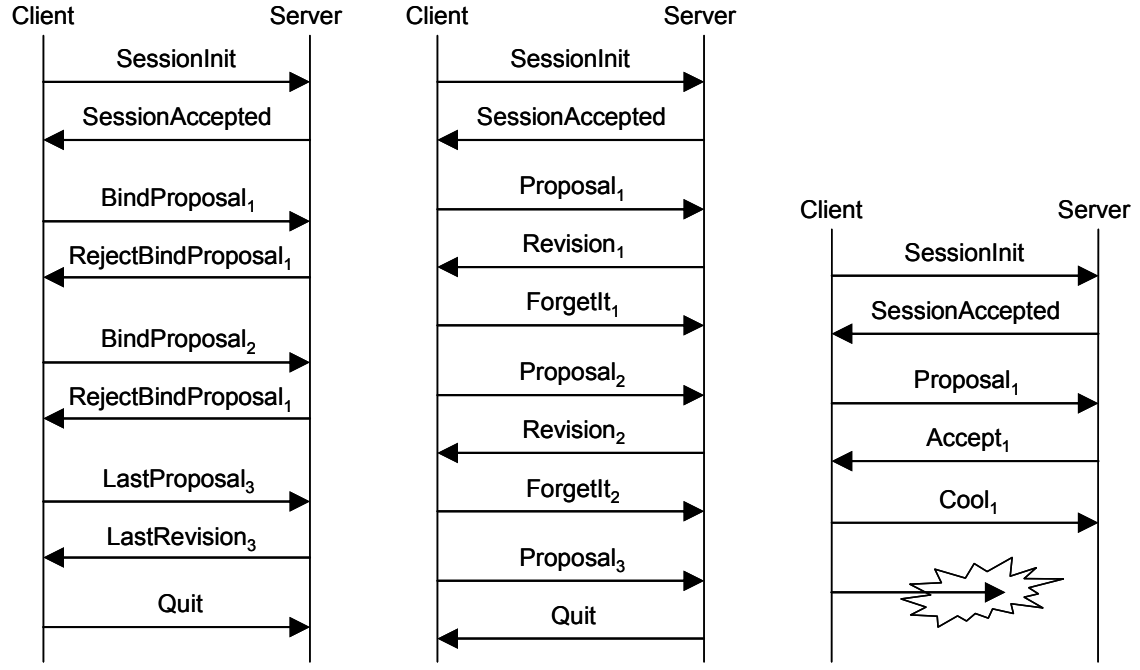


(a) Client explores different options before calling for agreement

(b) Server forces the client to conclude negotiations

(c) Client is granted extra time to explore more options

Figure 4-2: MSCs of successful negotiations.



(a) Client aborts negotiations after several failed attempts to establish agreement

(b) Server aborts negotiations without trying last round

(c) Transport layer failure, timers expire at boths sides and the protocol terminates

Figure 4-3: MSCs of unsuccessful negotiations.

Interface Messages

The interface that SrNP offers to applications realising negotiation logic is described in a technology-independent manner through a set of messages depicted in Table 4-3.

SrNP interface messages are self-explained and mainly correspond to the SrNP protocol messages presented previously. The prefix "Send" denotes the 'pull'-part of the SrNP interface allowing the application to send a protocol message to the other party, whereas the prefix "Forward" denotes the 'push'-part of the SrNP interface notifying the application of a protocol message received from the other party.

The `ForwardException` message notifies the application on abnormal protocol termination. This can occur (a) when the maximum tolerable time period that a party has specified to wait for the other party to respond elapses without receiving such a response, (b) on transport service failures and (c) on unexpected application behaviour. The latter occurs when the negotiation logic of a party responds to the last message received from the other party with a not valid response message or with multiple valid response messages.

SrNP Client Interface Message	SrNP Server Interface Message
SendSessionInit	SendSessionAccepted
SendBindProposal	SendAgreeProposal
SendLastProposal	SendRejectBindProposal
SendQuit	SendLastRevision
SendNewProposal	SendQuit
SendProposal	SendRevision
SendCool	SendAccept
SendForgetIt	SendReject
SendTime	SendTime
SendTimeGranted	SendTimeGranted
SendClarify	SendClarify
ForwardSessionAccepted	ForwardSessionInit
ForwardAgreeProposal	ForwardBindProposal
ForwardRejectBindProposal	ForwardLastProposal
ForwardLastRevision	ForwardQuit
ForwardQuit	ForwardProposal
ForwardRevision	ForwardCool
ForwardAccept	ForwardForgetIt
ForwardReject	ForwardTime
ForwardTime	ForwardTimeGranted
ForwardTimeGranted	ForwardClarify
ForwardClarify	ForwardProtocolException
ForwardProtocolException	

Table 4-3: SrNP interface messages.

4.3 Quality of Service Specification Language

4.3.1 Related Work in Specification techniques and Languages

In this section we give an outline of existing specification techniques and languages and show their relation to QoSSL and its underlying concept of a generic measurement process.

Most approaches in this area are bound to a specific application area (e.g. distributed service, grid services, or multi-media). UML-Q and UML-M [36, 37] provide extensions to UML to model QoS related stuff in distributed services. UML-M gives the basis to implement a measurement system. However, this implementation can not be generated and has to be created manually. QIDL [38] was developed as a part of the MAQS (Management for Adaptive QoS-enabled Services) project. It is designed to describe quality aspects of an interface (not of methods as QoSSL) in a CORBA base service implementation. Stubs to implement a measurement system can be generated with QIDL. The implementation of that system is not guided any further. An approach quite similar is done within ODL, the Object Definition Language [49]. QDL (Quality Description Languages) [48] were developed in QuO (Quality Objects, another CORBA-based framework). Specifications are strictly bound to distributed systems and include kind of conditional rules to define a systems reaction to QoS degradation. A QoS feature as determinable in QoSSL can not be specified as is. The QuO framework also offers a runtime environment to ensure the enforcement of the rules specified. However, it does not deliver values of the QoS measured because the QoS management is integrated in the framework. This hinders the application of QDL and QuO in existing (management) systems. QML (Quality Modelling Language) [42] was developed at HP Labs. Its main focus is to specify the QoS desired. Therefore QML and its associated run-time-environment QRR provide kind of quality enforcement but do not generate a measurement system as QoSSL does.

Very specialized approaches are carried out for Web-Services with WS-QoS [51] and for OGSA-based Grid-Services with G-QoSM [35]. HQML - Hierarchical Quality of Service Markup Language [46] and XQoS [40] primarily focus on multi-media systems. The IPPM [50] framework is strictly bound to IP but is based on a well structured approach of defining a common measurement process applicable to any QoS measure in the IP area. QoSSL adopts this approach while also broadening its scope.

Specification approaches explicitly dealing with any kind of service are for example given in [39] and [41]. The first one uses dependency graphs to describe QoS features in service hierarchies. As these graphs are hard to derive in real life systems its applicability is questionable. The second approach called QuAL (Quality Assurance Language) mainly focuses on assurance and not on measurement of QoS features.

Summed up, specification techniques existing today either do not support the whole service life cycle by automatically generating a measurement system or they are strictly bound to very specific services or mainly focus on assurance and not on measurement. QoSSL offers an approach broadly applicable strictly focused on measurement. Quality management is delegated to a discrete management implementation.

4.3.2 Specification on QoS

Provider and customer usually depend on the definitions of high level parameters to specify as well as to monitor their requirements on quality of services respectively. Instead of looking at those specific parameters, we start our investigation of QoS specifications based on service life cycle. This aspect of QoS specification is not thoroughly covered by current research. Then we introduce a formal model for the specification.

4.3.2.1 QoS Specification from the Life Cycle Point of View

In the following section, requirements on a QoS specification technique are derived from the point of view of the service life cycle. As a survey of specification techniques and languages as done in [45] and as we have shown in previous section, the aspect of life cycle orientation is not yet reflected in any specification concept. The requirements listed in this section form the basis for the design of the newly introduced specification language QoSSL.

The specification of the QoS that has to be hold during the operation of a service is one of the tasks accomplished in the negotiation phase of a service. This determination has a massive impact on the following phases of the service life cycle. To ensure that these agreements will be met their impact has to be kept in mind.

When a new service is planned in the negotiation phase neither an implementation of that service nor any information concerning the execution of that service (e.g. traces) is available. However, during the negotiation phase important decisions either triggered by QoS requirements or influencing the QoS of the service in the following phases have to be made. Even when dealing with standard services additional QoS-features may be agreed on in the negotiation phase.

If the service to be realized is built upon other sub services the problem of determining a service's properties before it is even implemented or executed intensifies because it reoccurs for every sub service which has to be negotiated upon. Analyzing QoS-specification from this life cycle oriented point of view yields the requirement for QoS-specification to be a reusable basis determining a service's properties in every phase of the service life cycle following the negotiation phase. This determination has to be done automatically so that further characteristics of a service (e.g. its implementation) may be derived automatically as well and are thus directly bound to the quality specification. This automatic derivation enables the forecast of the impact of the specification of QoS features on the remaining phases of the life cycle.

An automation process only based on specifications made in the negotiation phase therefore has to deliver the results described in the following to determine the impact of a QoS specification on the following life cycle phases. Figure 4-4 illustrates the derivation of these results in respect of the service life cycle. (The left third of this figure shows the typical hierarchy of services as introduced in [44]. Between the roles involved (customer, provider, sub provider) typical questions of service management already arising in the negotiation phase are displayed. The life cycle phases are plotted on the axis form the left to the right)

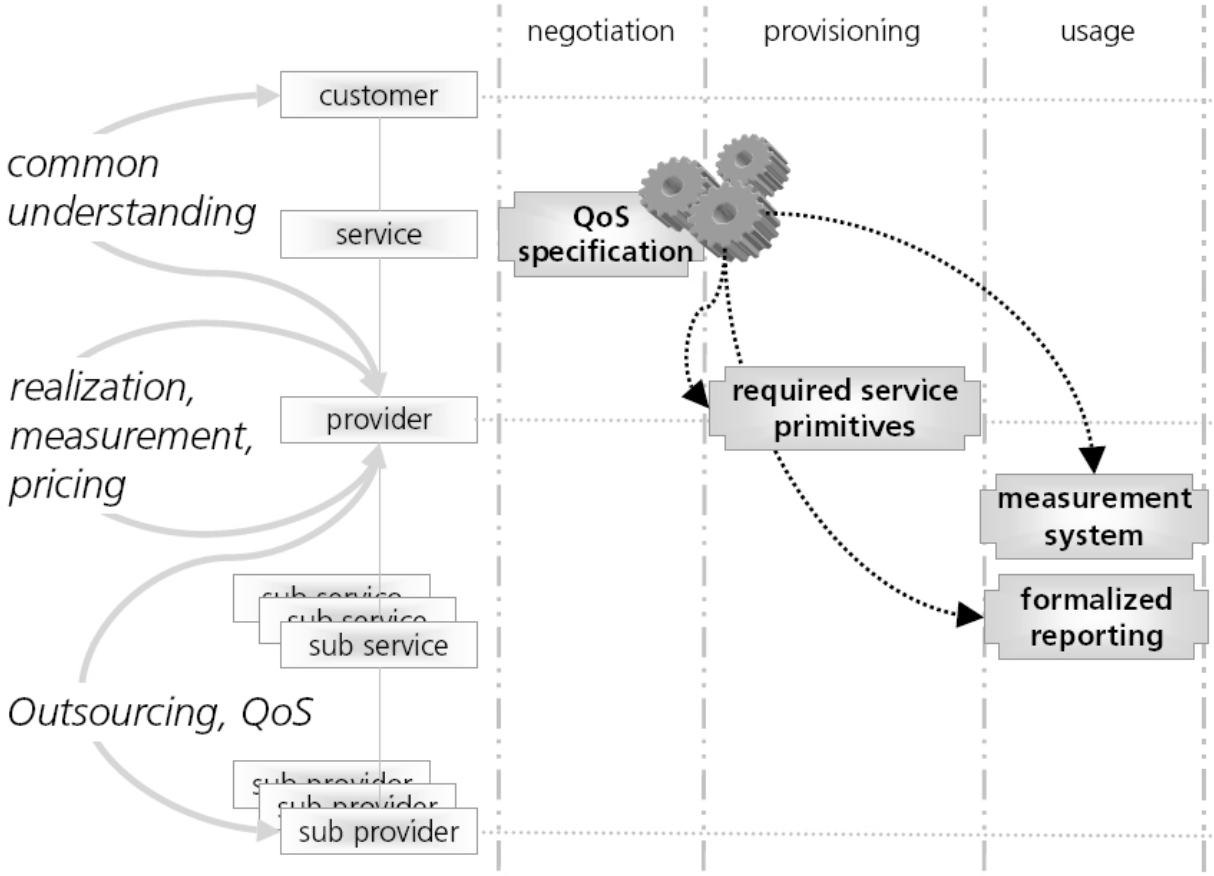


Figure 4-4: Basic steps calculating the impact of quality specification in the negotiation phase.

Required service primitives A specification of QoS laid down in the negotiation phase is not able to access any information of succeeding life cycle phases. Apparently, a specification technique independent of details of a service's implementation should be chosen to specify the QoS features. In the following we therefore assume that the specification of QoS is based on the analysis of the invocations of (abstract) methods since these had to be specified to determine the service's functionality anyway.

The detailed discussion of this specification technique as done in [45] emphasises the need for a suitable mechanism to intercept method invocations so that the values of the particular QoS features can be calculated at last. Thus it is necessary for a QoS specification to well define the methods it is based upon. Abstracting from the object oriented view on a service the invocation of methods can be seen as the call of a service primitive as introduced in the OSI layering concept [47]. Accordingly, one requirement is that the service primitives necessary to acquire a certain QoS feature have to be derivable from its specification. This definition of a service primitive corresponds to the definition of service access point as laid down in [43].

Measurement system During the operation phase of a service the QoS currently achieved must be measurable. Therefore a suitable measurement system has to be built already in the provisioning phase so that relevant data can be gathered during the operation phase. To implement a measurement system the properties determining the QoS must be accessible for that system. In the case of the approach of analyzing method invocations chosen here these invocations must be recognizable by the measurement system respectively.

Formalized reporting In order to be also applicable when integrating sub services a specification technique or language must provide means to exchange data concerning the QoS and its specification. Doing so, information about the QoS of a sub service can be provided without details about the respecting sub service being necessary.

4.3.2.2 QoS specification based on a formal model

In the previous section we claimed that all further details required in the following phases of the life cycle should be derivable of a sole specification laid down in the negotiation phase. The most complex derivable partial result is the implementation of a measurement system applicable in the operation phase.

Thus, it would be important to align the specification language to that measurement system so that it can always be successfully generated as constructed. Practically, the specification language is not aligned to one specific measurement system but to a generic and configurable measurement process which has as well to be designed based on a generic service model. This general measurement process will then be configured using a corresponding language statement so that the resulting measurement system is perfectly suited to measure the QoS feature specified in that language statement.

This configuration is automated and thus can be realized with computational support. Correspondingly, the remaining results specific to each phase of the life cycle are derivable of the specification and thus can be generated computer aided and could be shown in detail with following list

Negotiation phase A specification in QoSSL is designed to determine a QoS feature only with the information available in the negotiation phase. Therefore no further generations are necessary in this phase of the life cycle.

Provisioning phase The service implementation build in this phase has to be complemented with the implementation of a measurement system specific to the QoS features specified. That means a suitable configuration of the common generic measurement process is generated.

In addition, the interfaces the measurement system will interfere at with the service implementation have to be named so that these interfaces are integrated in the service implementation. The list of necessary interfaces is generated out of the QoS specification in QoSSL, respectively. Furthermore the measurement system is “tapped‟” to service implementation following a generated “manual”.

Usage phase The measurement system configured in the provisioning phase only has to be started in this phase (together with the service implementation itself). The “tap” to the service implementation is started as well. Doing so, a running measurement system delivering values of the QoS characteristic specified in QoSSL is available.

QoSSL as the basis for the steps of generation shown above is formalized as far as possible to be automatically processable and human readable as well. It builds a declarative language which is well suited to each generation step. A formal model describing a common generic measurement process is used as the development basis of QoSSL. This model is introduced in the next section before the syntax of QoSSL is shown in detail.

4.3.3 QoS Specification with QoSSL

4.3.3.1 Generic Measurement Process

The measurement process model we present here is an enhancement of the IPPM framework [50]. While IPPM is strictly focused on IP we reuse this concept with further considerations on service orientation. The IPPM approach is to intercept communication between systems, derive relevant events of this interception, correlate these events to form a so called sample, and aggregate these samples to a distribution representing the values of the QoS feature specified.

We adopt this scheme and generate events strictly service oriented by intercepting method invocations at a Service Access Point (SAP). These events are called QoS events and processed similar to the IPPM framework (correlation and aggregation). To merge this idea to already existing concepts in modelling services we integrated this general measurement process into the MNM service model [43] to clearly depict its relation to other parts and roles of a service as identified within the MNM service model.

The complete model (service and measurement process) clearly shows the distinction between classes exchanged between what we call activities (e.g. correlation, aggregation) and classes representing those activities. The first are represented as association classes describing association between the last. Figure 4-5 shows this model. Parts of the model not directly related to the measurement process are shown light grey to enhance visibility of the measurement process model. All classes except the tap class are designed as parts of the class QoS parameters already existing in the MNM service model as such.

Configuring the generic measurement process now means to derive a more specific model from the one shown in Figure 4-5. This derivation will be driven by a QoSSL statement. As we will show in next section the measurement processes presented here forms a universal basis to that derivation.

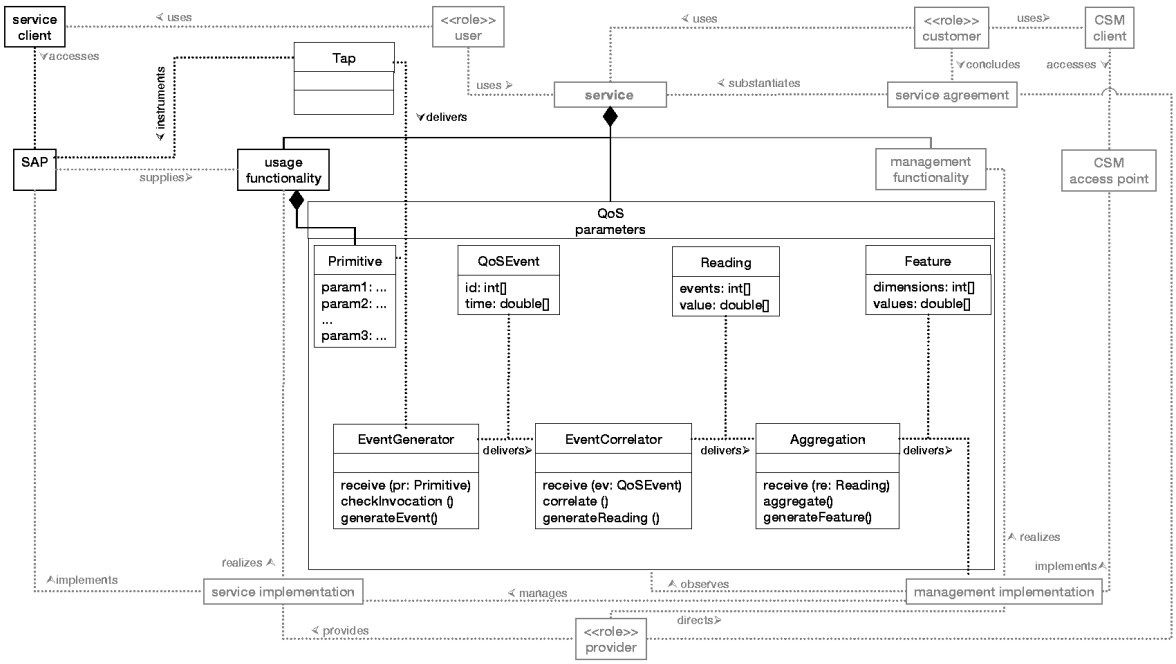


Figure 4-5: UML-class model showing the generic measurement process and its integration into the MNM service model.

4.3.3.2 QoSSL

Regarding our approach introduced in section 4.3.2, QoSSL is one possible definition of a language suitable to configure the generic measurement process introduced in above section. Together with QoSSL we have implemented a compiler transforming QoSSL into java classes corresponding to the measurement process model we introduced. Together with this compiler, QoSSL enables the computer aided specification and measurement of QoS features.

In the following, the language elements of QoSSL are described in same order as they would have to occur in a QoSSL "program" specifying a QoS feature. The formal definitions of QoSSL statements are given in BNF together with a short example statement.

4.3.3.2.1 Defining Primitives

```
PRIMITIVE> := "PRIMITVE" <Primitive_Identifier>
       ("("
         <Attribute_Identifier> ":" <type>)
         "," <Attribute_Identifier> ":" <type>)*
          "); ")?
```

Keyword	Comment
Primitive	Description of a service primitive together with its attributes.
Primitive_Identifier	This identifier is arbitrary, however a primitive should be named in accordance to the service specification.
Attribute_Identifier	Attributes can be specified using the type definition concept described in the next item.

The following example shows a definition of the UDP receive service primitive. (presuming an already opened socket)

```
PRIMITIVE recv ( data:     BYTE []
                 len:      BYTE [2]
                  );
```

4.3.3.2.2 Declarative Type Concept

```
<TYPE> := <BIT> <ARRAY_DEFINITION>
       | <BIT>
       | <BYTE> <ARRAY_DEFINITION>
       | <BYTE>
       | <CHAR> <ARRAY_DEFINITION>
       | <CHAR>
```

Keyword	Comment
TYPE	The version of QoSSL introduced here supports simple types like BIT, BYTE, and CHAR only. Attribute may be fields of arbitrary length to represent payload for example. An extension of this simple type concept is possible and necessary to represent service primitives on higher layers (e.g. application layer). Currently, we are integrating the whole java type concept into QoSSL so that attributes of a service primitive may be represented as complex objects as well.

4.3.3.2.3 Defining QoS Events

```
<Event> := "EVENT" <Event_Identifier>
          "USES"   <Primitive_Identifier>
          "ON TAP" <Tap_Symbol>
          "ISOLATE" "(" (<Filter_Term>)
                   ("," <Filter_Term>)* ")"
           "ID"    "AUTO"
                 | <Attribute_Identifier>
                  | <Hash>
          ";"

<TAP_SYMBOL> := [1. .9] [0. .9]
<Filter_Term> := "*"
               | ( <COMPARE>
                   ( <NUM_CONSTANT>
                     | <ALPHA_CONSTANT>
                     | <IDENTIFIER>
                   )
                 )
               | ( <EVAL_FUNCTION> <COMPARE>
                   ( <NUM_CONSTANT>
                     | <ALPHA_CONSTANT>
                     | <IDENTIFIER>
                   )
                 )

<EVAL_FUNCTION> := "size"
<Hash> := "HASH" "(" <Attribute_Identifier>
             ("," <Attribute_Identifier> )* ")"
```

Keyword	Comment
EVENT	Starts the definition of an event relevant to a QoS feature or QoS event in short. It has to be named with a unique identifier.
USES	Identifies the service primitive that this QoS event is based on.
ON TAP	The tap instance is responsible to intercept primitive when the service is running. It is symbolically named after this keyword. This symbol has to be reused when a tap is implemented.
ISOLATE	Filters the service primitive interceptions so that only invocations relevant results are in a QoS event. A filter term is placed after this keyword. Therein a filter rule has to be specified for every attribute of the service primitive that the QoS event definition is based on. Rules may contain calls to functions or constants.
SIZE	Return the length of an array
ID	Determining the numeric ID of an QoS event instance specified. Every QoS event has to be uniquely identifiable to ease the correlation later on. This term may simply be an attribute ID referencing an attribute of the service primitive or a more complex calculation using even hash functions to uniquely identify the payload
*	The wildcard * can be used if no filtering is required.

The example following shows a definition base on the *data* primitive of the previous example. The filter term is designed to only generate QoS events when the payload length is greater than 64. The resulting QoS events are uniquely identified using a hash of the attributes *dest* and *data*.

```
EVENT udp_foo USES  recv
            ON TAP 1
            ISOLATE (*,*,*,*,size > 64)
            ID      HASH(dest,data);
```

4.3.3.2.4 Determining a Reading (Measure)

```
<Reading> :=
  "READING" <Reading_Identifier>
  "REQUIRES" <Event_Identifier>
             "AS" <Queue_Identifier>
           ("," <Event_Identifier>
             "AS" <Queue_Identifier>) *
  "COMPUTES" <Term>
  "UNIT"     "bit" | "bit/sec" | "sec" | "%" | "NULL"
   ";"
```

Keyword	Comment
READING	Correlates multiple QoS events to build a single measure or reading (named sample in IPPM [19]).
REQUIRES	Specifies the classes of QoS events required to calculate this reading. The amount of event definitions following this keyword is not restricted.
AS	The single instances of QoS events reaching the event correlator are buffered in queues which have to be explicitly named after this keyword.
COMPUTES	Defines computation rules which actually perform the event correlation.
UNIT	Specifies the unit of resulting reading.

An example definition is given after introducing the correlation statements in the next item.

4.3.3.2.5 Correlation Rules

```
<Term> := <numGivingEventFunction>

       | <numGivingQueueFunction>

       | <Term> <OPERAND> <Term>

       | "(" <Term> ") "

       | <CONSTANT>

<numGivingEventFunction> : =<Event_Identifier>

                          |<eventGivingFunct ion>

                          "."

                         ( "time" | "id" )

<numGivingQueueFunction> : =<Queue_Identifier>

                          "."

                         ( "count" )

<eventGivingFunct ion> : =<Queue_Identifier>

                            "."

                         ("mymatch" | "hismatch")

                          "(" <Queue_Identifier> ")"
```

Defining the computation of reading - the correlation of QoS events - is the most complex definition of a QoSSL program when defining a QoS feature. Correlation function

(only a few are used in the prototype presented here) can arbitrarily be mixed up and combined with typical mathematical operands.

Three different types of correlation functions are available in QoSSL.

Type of Correlation Function	Comment
`numGivingEventFunctions`	Those functions are directly applied to QoS events and instantly returning a value.
`numGivingQueueFunctions`	Those functions are applied to queues and return a value.
`eventGivingQueueFunctions`	Those functions are applied to a queue and returning a distinct instance of an QoS event.

For simplicity in compilation all function are monadic. Further correlation function may be added as long as they respect this classification determining possible signatures of the functions. In our prototype we support a minimal set of correlation functions.

Correlation Function	Comment
`Time`	This function is applied to QoS event instances and returns their time stamp.
`Id`	This function is applied to QoS event instances and returns their ID.
`Count`	This function returns number of events contained in a queue.
`mymatch` and `hismatch`	They return a pair of events with the same id contained in both queues. One out of the queue the function was called at (`mymatch`) the other one out of the queue which was the argument to the function call.

Computation of a reading is only begun if events are contained in every queue specified after the `REQUIRES` keyword. If the computation was successful (that means every function call following `COMPUTES` returned a value) all events used for this computation are de-queued. Thus, every time suitable event to correlate are contained in the queues a reading will be returned.

The example following presumes a definition of the three QoS events `tcp_syn_sent`, `tcp_syn_ack_received`, and `tcp_ack_received` in a way that they represent the service primitives necessary to call, when setting up a TCP-connection. The definition of the time passing to set up a connection (connection establishment time) is shown below. A further example of QoSSL program will be given in next section when the compilation of a definition resulting in configuring the generic measurement process is shown in detail.

```
READING tcp_cet_reading
       REQUIRES tcp_syn_sent                AS a,
                   tcp_syn_ack_received       AS b,
                   tcp_ack_received               AS c
                COMPUTES a.myMatch(c) .time
                                      - a.hisMatch(c) .time
                UNIT ms;
```

4.3.3.2.6 QoS Features

```
<FEATURE> := "FEATURE" <Feature_Identifier>
            "JOINS" <NUMCONST> <Reading_Identifier>
             "TO"    <Aggregation>
```

The final step to complete the specification of a QoS feature is to specify how single readings have to be aggregated to represent a QoS feature. (As in IPPM we argue that a service's QoS is not represented by single values but aggregations thereof like distribution functions or box-plots and so on). These aggregation functions are described in the following section.

Keyword	Comment
FEATURE	Names the QoS feature.
JOINS	Aggregates the number of reading and denotes the name of that reading.
TO	It determines the update frequency as well as controls the number of readings delivered to an aggregation function.

4.3.3.2.7 Aggregation Rules

```
<Aggregation> := "SUM"
                 "HISTOGRAM"
                 "DISTRIBUTION"
                 "BOXPLOT"
                 "MOVING AVERAGE"
```

Possible functions to aggregate readings are displayed in the syntax definition above. The functions enlisted in the syntax definition have to be implemented when compiling a QoSSL program. The following example shows the definition of the QoS feature moving average of 10 sample of connection establishment time in TCP".

FEATURE tcp_connection_time

JOIN 10 tcp_cet_reading TO MAVG;

As can be seen in the statements shown in this section, QoSSL is a simple and declarative language to specify QoS features. The power of QoSSL results from automated compilation (in our prototype to java). To ease this, we developed a compiler as well as an runtime environment. To achieve a compound presentation, we do not go into details here but present an example QoSSL program and its compilation in the next section.

4.3.3.3 Implementing QoSSL

In this section we discuss an example QoSSL "program" and its realization to give an insight to the design of QoSSL, the generic measurement process, and the QoSSL compiler. The specification of the frame error rate in Ethernet is given as follows

```
PRIMITIVE recv# ( data:     BYTE[]
                  crc:      BYTE [4]
                  );

EVENT recv_ok USES  eth_recv#
                    ON TAP 1
                    ISOLATE (* , ==crc (data))
                    ID   AUTO;

EVENT recv_faulty USES  eth_recv#
                        ON TAP 1
                        ISOLATE (*, !=crc (data))
                        ID     AUTO;

READING eth_faulty_frame
       REQUIRES recv_ok       AS a,
                recv_faulty    AS b,
                COMPUTES b.count / a.count * 100
                UNIT %;

FEATURE eth_faulty_frame_rate

JOIN 10 eth_faulty_frame TO MOVING AVERAGE;
```

This definition is based onto one sole service primitive (the reception of an Ethernet frame) only. If the CRC contained is equal to a CRC locally computed an QoS event is generated representing the correct reception of a frame. Another QoS event representing a faulty reception is generated if both CRC do not match respectively. The QoS events are correlated by computing the ratio of their occurrence. The values are aggregated to form a moving average of 10 readings.

Figure 4-6 shows the refined model resulting from the configuration (actually a refinement) of the generic measurement process introduced in the above section. For a comprehensive presentation we leave out classes of the MNM service model here and show classes building the generic measurement process in light grey. The following detailed description of that model is ordered in accordance to the processing steps of the generic model to explicitly show the refinement taking place.

Tap

The class *Ethernet_Tap* is generated as a derivation of the class Tap. As tapping to service implementation can not be done automatically the class *Tap* as well as the class *Ethernet_Tap* only provides a stub for implementation also offering means to connect to the specific event generator.

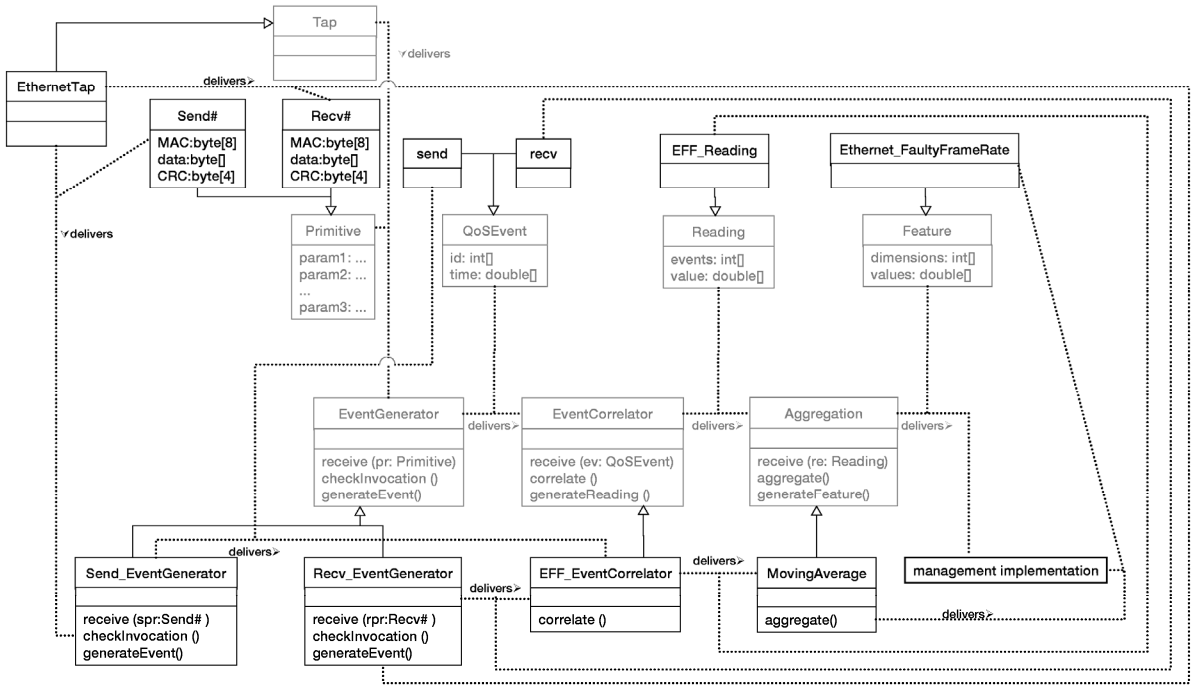


Figure 4-6: Refined class model to describe the rate of incorrectly received frames on an Ethernet connection

Primitive

The class *Primitive* is a container to specify the data exchange format between a tap and an event generator. According to the specification done in QoSSL a class *recv#* is generated as a refinement of *Primitive*. This refined class is bound to the association delivers between the refined tap and the refined event generator. (In our java implementation this binding is done by specifying suitable interface at taps and generators and providing means to register event generators to receive primitives at taps.)

Event generator

The *Recv_EventGenerator* is implemented with refining the generic *Event-Generator* and overloading its methods according to the QoSSL specification. Filter terms determine the implementation of the *checkInvocation* method. The interface of the *receive* method is adopted to be able to receive *recv#* primitives. A specialized event generator is build by the compiler for every QoS event specified in the QoSSL statement. In case of our example this results in the two classes *Recv_EventGenerator* and *Fault_EventGenerator*.

QoS event

Refinements of *QoSEvent* are generated for each QoS Event specified in QoSSL after the `EVENT` keyword. Accordingly, the two classes *recv* and *fault* are generated, both refining *QoSEvent* and being bound as association classes to the association between the specialized event generator and event correlator (implemented in java similar as described with the refinements of *Primitive*). As we implement a lightweight event concept here, event only carry a time stamp and an id as attributes. Thus, further refinement than defining a new name when building specific events is not necessary

Event correlation

One specific event correlator is derived from the class *EventCorrelation* for every QoS feature specified in QoSSL. In the example the class *EFF_EventCorrelation* is a refinement of *EventCorrelation* simply overloading the method *correlate* with the correlation function determined in the corresponding QoSSL statement. (In the java implementation correlation is implemented using a set of predefined correlation functions implemented in java which are simply composed by the compiler according to the QoSSL specification.)

Reading

Similar to the other classes representing data structures the class *EFF_Reading* is a simple derivation of *Reading* only representing a new name and clarifying interfaces. The attributes of the upper class Reading are useful to any reading and thus do not need further refinement.

Aggregation

A refinement of *Aggregation* is chosen according to the QoSSL specification with the keyword `TO` in the `FEATURE` statement. (In our implementation these refinement are generating using an already implemented repository of aggregation functions represented as classes with a common interface.)

Feature

Every feature defined in QoSSL can be represented with the class *Feature* in the common model. For clarity a class specific to the specified feature is derived to introduce a clear name (in our case the class *EthernetFaultyFrameRate*). This refined feature is bound to the association which links the aggregation and the service management implementation together indicating that this feature is "delivered" to the service management implementation. (Implementing this binding in java means that the service management implementation must register itself the specific aggregation (*MovingAverage* in our case) to receive this feature. This registration can not be done by the compiler but has to be implemented "by hand" since the service management implementation is not known to the QoSSL compiler.)

4.4 Conclusions

In this section we presented an architecture for automated QoS management for fixed IP networks. We described in brief the functional components that have to be involved in the process, ranging from the service negotiation phase to the service provisioning and deployment and we focused on the interface language/protocols involved in the automation of the negotiation phase and in the automation of the configuration of the QoS monitoring system of a provider.

In the future we will elaborate on the functionality of the blocks that were only briefly and in a high level described in this document.

5 Autonomic Management of Ad hoc Networks and Ubiquitous Environments

There exists an apparent and increasing interest towards ubiquitous environments in order to provide services anytime, anywhere, from any devices. Ad-hoc and sensor networking can be viewed as a real life realization of such environments. In ad hoc networks, mobile nodes are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Conventional wireless networks require some form of fixed network infrastructure and centralized administration for their operation. In contrast, since ad-hoc networks are self-creating, individual nodes are responsible for dynamically discovering other nodes they can communicate with. This way of dynamically creating a network often requires an equally dynamic ability to rapidly initialize, deploy and manage services and protocols in response to user demands, higher-level management goals and surrounding conditions.

The management of ubiquitous environments is definitely harder and more complex than the management of traditional fixed environments. Indeed, ubiquitous environments are highly dynamic and are typically self-maintained by mobile devices that operate under many constraints such as bandwidth limitation, energy exhaustion, and low system capacities. They provide new challenges towards managing them. Autonomic management is a promising area of research in this context and covers four functional areas: self-configuration, self-healing, self-optimization and self-protection. The objective is to model, design and experiment new management architectures and protocols to provide to these environments the capabilities to manage themselves and to overcome their changes in a dynamic and autonomous manner.

This section synthesizes the research efforts in the area of autonomic management for ubiquitous environments, performed at the Imperial College, UK, at the University of Surrey, UK and at LORIA/INRIA, France. In this part of the deliverable, we address two major issues: (1) the reorganization of the management plane and (2) the design of two autonomic functions: self-configuration and self-healing.

5.1 Special characteristics/requirements of Ubiquitous Environments

Ubiquitous Computing Systems (UCSs) will consist of millions of potentially inter-linked computers embedded in clothing, people, animals, vehicles and buildings, in the countryside and in the atmosphere. The described Ubiquitous Environment will include an infrastructure of networking and major computational facilities supporting wired and wireless communications, processing and storage. People will also be a part of UCSs, some concerned with their operation, others just as users. The UCS could, for example, help them to perform their normal daily activities, monitor their medical problems, entertain them and support collaboration and interaction between them.

The potential applications include healthcare by monitoring chronically ill and post-operative patients in hospitals and facilitating early release to be monitored in homes. Closed loop control using therapeutic actuators such as cardiac defibrillator, nerve stimulation and drug-delivery will be feasible. Monitoring in the home will not be confined to chronically ill but for healthy people who are at risk due to possibly inherited gene defects. In addition elderly or disabled people could benefit both from monitoring and a UCS which assists them go about normal activity. Other applications include

environmental monitoring for climate change, bad weather warnings, earthquake warnings etc. A sensor network monitoring water flows and levels along rivers could provide early warnings about floods. Interactive entertainment, commerce and education will become available everywhere and at any time. These systems will have to support ad-hoc collaborations for applications such as emergency services attending a disaster site or between autonomous unmanned vehicles collaborating on a military mission.

Billions of entities – devices, sensors, mobile phones, software components, databases, people etc will be part of future UCSs. They will form dynamic communities for specific purposes. Many will be mobile and error-prone and some will be malicious and try to harm other entities around them. Such a fluid, dynamic system has to be *autonomic* – self-managing, self-configuring, self-healing, self-protecting – but must cater for human intervention where appropriate to set preferences or override autonomic decisions where appropriate. Yet the subject of autonomic systems is in its infancy, as yet far from capable of dealing with systems of the envisaged complexity, or with the range of threats to their dependability and security that are likely to arise in practice.

The characteristics of a ubiquitous computing system are potentially billions of devices and sensors producing vast quantities of information. Many elements are mobile and communicate via wireless links. A UCS is generally considered context-aware in that it can adapt to current context – a concept which covers more than just location, since it can include available resources such as battery power, device capabilities, wireless bandwidth, the presence and activity of both humans and other entities within the UCS. The UCS will be very dynamic as entities are both mobile and adaptive but its introduction is likely to modify the behaviour of the humans which will use and interact with it, causing it to further adapt to the human activity. The management of ubiquitous/ad-hoc networks is challenged by several constraints that are not encountered in the common fixed networks:

- Relevant management information: the challenge is to identify the essential pieces of relevant information but also to determine what management operations to be taken in this context. While we have a fair amount of understanding about what monitoring information should be collected, a few work focus on what management actions to be taken and on the demonstration of their effectiveness.
- Management domain related: since ad-hoc networks are dynamically formed loosely coupled entities, the notion of administrative domain, responsible for the overall management can be challenging for some application domains. Even if we consider an hybrid ad-hoc network connected to the gateway of a company, cooperative and role-based schemes should be further investigated to organize the management plane in an efficient manner.
- Cost of monitoring: resource consumption due to management is neglected in fixed networks, while the same is of major importance in a landscape where bandwidth and battery lifetime are the key actors. Resources (battery power) are already scarce in ad-hoc networks and management actions are additional consumers in an already resource under-powered context. Defining lightweight schemes, where already available information is used and resource availability and consumption can be minimal, are the main issues.

Management node reliability and willingness to cooperate: an agent/manager node is limited by its inherent limits (connectivity, failures) such that the provided management data may be biased. How to improve the reliability of management data which are provided by other nodes, as well as the reliability of data provided by the node itself? Additionally, malicious nodes or non-cooperative nodes may provide false data or no data at all. The key issue is to define distributed monitoring and management paradigms, where biased or faulty information can be out-weighted or discarded.

5.2 Approaches towards self-management

Future networked systems will include sensor networks used for monitoring the environment, buildings, and personal health. These may potentially involve millions of devices so manual configuration and management is impractical. Wired networks and distributed applications are becoming increasingly complex and difficult for operators to manage. These complex systems need to be able to configure themselves with little or no user input, but more importantly, it is required to adapt autonomously to changes such as user movement, device failure, and the addition or loss of services. Ubiquitous environments are an emerging and extremely promising concept. Inherent benefits though such as unrestrained computing, lack of centralization, ease of deployment at low costs, are tightly bound with their deficits in fields such as lack of controlled management and limited resources. There is a need for management frameworks to handle the complex requirements posed by this networking paradigm.

We have identified three frameworks aiming towards the management of Ubiquitous Environments. The first approach is based on the concept of Self-Managed Cells (SMCs), targeting lightweight devices like sensors, PDAs as well as network elements such as firewalls and routers. The next approach introduces a probabilistic approach to provide guarantees of managing certain percentages of mobile nodes. This approach is targeted towards MANETs but the concept is applicable to Ubiquitous Environments, where a plethora of nodes need to be managed. The final approach presented employs a Policy-Based Network Management approach in order to manage Mobile Ad Hoc Networks. Based on a hybrid organizational model, this approach distributes policies among the clustered nodes which automatically enforce those to all MANET nodes, leading to a degree of self-management and self-configuration.

5.2.1 Self-Managed Cells

We advocate the concept of the Self-Managed Cell (SMC) as an approach for implementing autonomic ubiquitous systems. An SMC manages a set of components such as those in a body-area network, a building, a set of mobile autonomous vehicles or even a large-scale distributed application. The components could be sensor nodes, smart phones and PDAs as well as network elements such as firewalls and routers.

The main objective of an SMC is to support autonomic functions such as self-configuration, self-healing, self-optimization, self-protection and context aware adaptation as appropriate. In many applications, users would not have the technical knowledge to configure the devices e.g. for healthcare a medic would assemble the devices needed to monitor a patient. Instead the SMC should discover appropriate components and configure them to perform the required role within the SMC. Similarly, if a failure is detected by some components, the system should reconfigure to continue working, although possibly with some reduced functionality.

There are several core services required for an SMC as shown in Figure 5-. In a very simple system such as a body-area network, we assume these are implemented by one of the nodes, which has processing and communication capabilities similar to that of a smart phone.

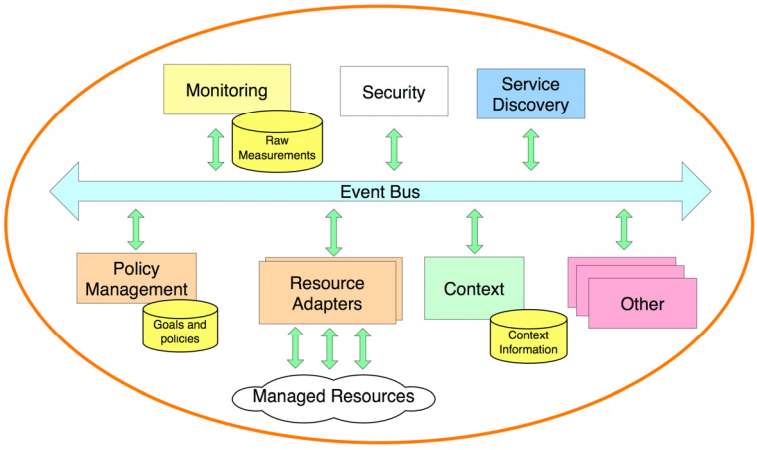


Figure 5-1 SMC Architectural Pattern

Event Bus: adaptive ubiquitous systems are essentially event driven as changes of state in resources need to be notified asynchronously to several, potentially unknown, recipients. An event may indicate discovery of a new component, component failure, a security attack and change in context or may be an application event. We are implementing a simple publish-subscribe event system supporting at-most-once persistent event delivery in which the service attempts to deliver the event until it knows that the subscriber is no longer a member of the SMC.

Discovery service: it is essential to discover nearby components which are capable of being members of the SMC e.g. intelligent sensors, and other cells when they come into communication range. The discovery service interrogates the new devices to establish a profile describing the services they offer and then generates an event for other SMC components to use as appropriate. We have to cater for mobile wireless components which may drift in and out of communications range and distinguish this from permanent departure from the cell.

Policy Service: policies provide the means of specifying the adaptation strategy for autonomic management in response to changes of state in the managed resource or changes of context in the environment in the form of event-condition-action rules. In essence, a SMC is a “closed-loop” system where changes of state in the resources trigger adaptation which in turn affects the state of the system. There is also the need for authorization policies to specify what actions a component within the cell can perform and what resources and services should be made available to external components. Policies can be added, removed, enabled and disabled to change the behaviour of cell components without reprogramming them. Policies also govern the behaviour of all the other components such as discovery service, context, security and even the policy service itself, allowing these to be tailored to specific applications. Ponder2 [67] is an implementation of a policy service specifically built for this purpose.

Roles: provide a means of grouping components according to the *role* they play within the cell e.g. temperature sensors, heart sensors, medic interacting with the patient in a healthcare scenario. Roles in a network could correspond to core routers, edge routers, firewalls, gateways, operators etc. When a component is discovered and authenticated,

it will be assigned to a role based on its capabilities and credentials. A set of policies relating to the role will then be loaded onto the components to govern how it interacts with other components in the cell.

Context: context-awareness enabling adaptation to current context is a key characteristic of most ubiquitous systems. This implies that sensors are needed to detect context which could include current location, user activity, weather, other entities in the vicinity etc.

Security: is essential in many applications. Components discovered by the discovery service will have to be authenticated and there may be a need to distribute encryption keys to newly joined SMC members to enable confidentiality. As cells may be mobile, the security may have to adapt to current context or when an attack is detected.

Resource adaptation: there may be a need for device specific adaptors to perform communication protocol conversion or device specific actions to present a consistent interface for the managed resources within a cell. This could also orchestrate the monitoring of resources related to performance, usage etc.

This list is not intended to be exhaustive and in the ubiquitous internet environment other management services such as the following may also be present.

Analysis/Simulation/Optimization: a component that analyses the measurements received from the managed resources and performs policy actions such as re-provisioning, operator alerts, etc

Accounting: a component that enforces the accounting policies of the network based upon the measurements received

5.2.2 Probabilistic management of MANETs

We proposed in [52] a new management approach for ad-hoc networks based on probabilistic guarantees. Instead of addressing the management of the whole network, we defined a scheme where a subset of nodes is managed in order to provide light-weight and reliable management. These nodes are determined based on their network behaviour to favour subsets of well connected and network participating nodes. With respect to such a selective management scheme, we derive probabilistic guarantees on the percentage of nodes to be managed. Our contribution is centred on a distributed management self-organizing algorithm at the application layer, its efficient deployment and a comprehensive simulation study.

Concept

The key assumption of the probabilistic management approach was motivated by the simple observation that ad-hoc network management should not comprise the totality of network elements (as it is typically the case in fixed networks). If node behaviour can be better described probabilistically (a node might be on-line with a given probability), why should we preserve a deterministic framework for the network management?

The first paper describing conceptual management approaches for ad-hoc networks was published by Burgess in [53], where a network model for ad-hoc networks is defined as a probability matrix, where an element in row i and column j is equal to the probability of nodes i and j to be directly connected. Our work is partially motivated by that paper: if we assume that such a matrix is known, why not restrict the network management plane to the nodes that seem to intercommunicate and have an effective network presence and define a probabilistic management scheme? The probabilistic

nature of the management plane derives from the guarantees that we can offer. We will not guarantee in this framework that all nodes will be managed, but we can give probabilistic bounds on the percentage of nodes that are managed on average.

We intend to determine a subset of nodes taking part into the management plane, such that we capture the most interesting nodes into the management. The notion of interesting is defined with respect to relative good network presence and topology relationship. We define a spatio-temporal component to be a subset of the network nodes, such that any pair of nodes has a high adjacency probability. The term spatio-temporal is inspired by the spatial dimension – nodes are required to be adjacent – and the temporal one – nodes should have a high probability to be adjacent. Since this probability is estimated over a given time period, we identify pairs of nodes that tend to directly interconnect. We restrict the management plane to the largest connected components, assuming that the nodes worth to be managed are located in these components. Since these components are associated to the largest node subsets having significant network presence and frequent adjacent relationships, we consider it is a viable approach for the specifics of ad-hoc networks.

Management algorithmic method

The algorithmic method for probabilistic management is based on the spatio-temporal connectivity measure. A network node evaluates its spatio-temporal connectivity with its neighbourhood and communicates that information to the other network nodes in order to construct the spatio- temporal connectivity matrix of the ad-hoc network. From this matrix, the node is capable to detect spatio-temporal connected components of the network and to elect its network manager. We assume a minimal cooperation among nodes, where partial control is allowed.

Spatio-temporal connectivity measure. An ad-hoc network is seen as a set of n mobile nodes $V = \{v_1, v_2, ..., v_n\}$ moving in a given surface during a time period T. The time period T is split in k measurement interval $[t_l, t_{l+1}]$ with $t_l = l \times T$ for an integer l in [0, k]. The choice of optimal time period T and interval values $[t_l, t_{l+1}]$ is not addressed here but an excellent analysis on timescales for information flow to monitor ad-hoc networks can be found in [54]. Each node v_i measures the percentage of time p_{ij} it was neighbour with a network node v_j. On a time interval $[t_l, t_{l+1}]$, the measurements are locally stored in a list $N^l_{stc}(v_i)$ composed of tuples (v_i, v_j, p_{ij}) and are subsequently exchanged and merged among network nodes. The suffix l represents the time factor and means the measure was performed during the time interval $[t_l, t_{l+1}]$. The exchange of local measurements lead conceptually to a network spatio-temporal connectivity matrix M^l_{stc} (or at least a partial view) obtained by concatenating the list of measurements performed by the network nodes. We will present how this exchange can be performed using a piggybacked routing protocol. Rows/columns stand for a network node. The i-th row of M^l_{stc} represents the list of measurements $N^l_{stc}(v_i)$ of a node v_i. If node v_i was neighbour with node v_j on $[t_l, t_{l+1}]$, an entry $M^l_{stc}[i, j]$ exists and contains the percentage of time p_{ij} that the pair (v_i, v_j) was directly connected on that time interval. As the goal is to highlight network nodes presenting a high probability of adjacency (and also to limit the management data), only the spatio-temporal connectivity values which are higher than a threshold value λ are considered in the matrix.

Connected component extraction. The matrix M^l_{stc} can be represented as a graph of n nodes as shown in Figure 5-2. A graph edge exists between two nodes, if the following double condition is respected: the two nodes were neighbours on the time

interval $[t_l, t_{l+1}]$ and the percentage of time p_{ij} is higher than the predefined threshold. The graph link is weighted by the percentage of time. Each network node v_i determines the connected component (noted CCv_i) of which he is part by running through the graph. A connected component corresponds to a subset of well-connected nodes in V. The CCv_i set is initialized with a single element: the node v_i itself. The key idea of the graph component extraction is to add recursively to the set CCv_i the neighbour nodes of each node already part of CCv_i. A node v_i can extract the connected component CCv_i, but can also extract the other components CCv_j he is not part of, by initializing a set CCv_j with a node which is not already assigned and by following the same approach. Each element of a resulting connected component has been a neighbour of another element during at least a minimum percentage of time.

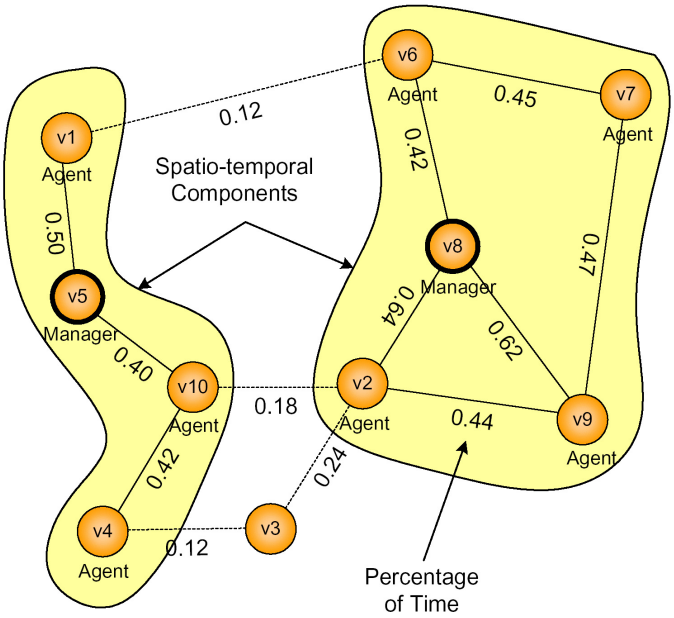


Figure 5-11 Extraction of spatio-temporal connected components

Routing protocol piggybacking. We introduced a piggybacked version of OLSR (Optimized Link State Routing Protocol) [55] in order to evaluate the spatio-temporal connectivity. As in a pure link state algorithm, each node performs two main operations: (1) it determines the list of directly-connected neighbour nodes by accomplishing link sensing through periodic emission of hello messages and (2) it exchanges this link state information with the other nodes throughout the network by flooding topology control messages. Due to the limited bandwidth of ad-hoc networks, OLSR reduces the control overhead using a specific heuristic based on Multi-Point Relays (MPRs) which forward broadcast messages during the flooding process. During the link sensing operation, an OLSR node uses a neighbourhood base (see RFC 3626 section 4.3) and maintains the list of one-hop neighbour nodes by recording a set of tuples (N_neighbor_main_addr, N_status, N_willingness) describing neighbours. The element N neighbour main addr stands for the main address of a neighbour, N_status specifies the link status (symmetric or not) with the given neighbour and N_willingness is an integer specifies the willingness of the neighbour to carry traffic on behalf of other nodes. We introduce an additional entry in the neighbour tuple called N_connectivity_time_perc to maintain the percentage of time the neighbour was connected to the node. This new entry must then be integrated in the topology control (TC) messages (defined in RFC 3626 section 9) performing the advertisement of link

states, so that the spatio-temporal connectivity information is propagated in the ad-hoc network. Including a sequence number, the format of a TC message describes a set of *Advertised Neighbour Main Address*. We define an additional field *Advertised Neighbour Connectivity Time Percentage* to include the percentage of time the neighbour was seen. Our spatio-temporal clustering algorithm extension, defined over a piggybacked OLSR version, gives to the ANMP architecture the capability to perform probabilistic management.

Experimental results

We performed a set of simulations, in which we apply the management algorithmic method to detect the largest spatio-temporal connected components. The simulations were done using the discrete event network simulator ns-2 [57]. We consider an ad-hoc network composed of $n \in [5 - 30]$ mobile devices implementing the OLSR routing protocol. The nodes are moving in a squared surface with a 1000 meters side according to the random way-point (RWP) model (being aware of the model limits [59]).

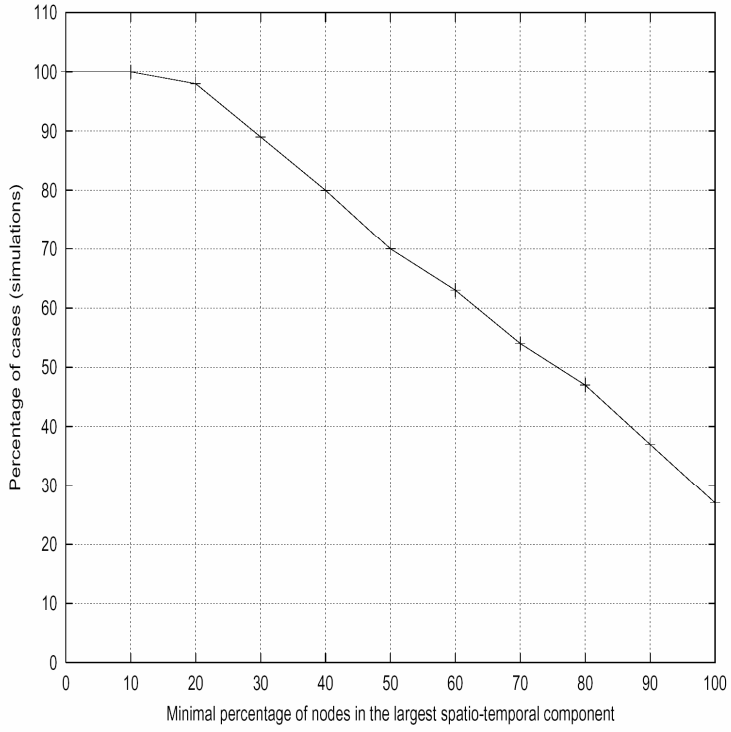


Figure 5-12 Probability distribution for the largest spatio-temporal connected component

In a first series of experiments we analyzed the probability distribution of the ratio of the largest spatio-temporal connected component to the overall network. These results are shown in Figure 5-3 and represent an extensive set of simulations with different mobility parameters and node sizes. For each individual setting, we performed 150 simulations to assure the non-bias of the result. On the x axis is plotted the percentage of the overall node size which is located in the largest spatio-temporal component, while the y axis stands for the percentage of cases (simulations). A point (x,y) on the graph stands for the percentage of simulations (y) such that the largest spatio-temporal component was at least x. For instance we can observe that the probability (y axis) of having at least 20% of nodes in the largest component is about 95%. We could have a best-effort type pragmatic management approach in the style: we manage around 20% of the nodes and in this case we will meet this requirement in

95% of the cases. We can assure the management of the half of the nodes within one single component with a probability close to 70%, which is a relative good result. Such a best-effort type of management could be implemented by one and only one manager station following the spatio-temporal component.

A natural question is whether network mobility impacts the management framework. Intuitively, higher mobility should make things worse (i.e.: smaller components sizes for the same probability), but a precise quantification of this effect is required. A second series of experiments addressed this issue, where different mobility parameters were evaluated for different node sizes. For each node size, 150 simulations were performed to avoid any bias. These results are summarized in Figure 5-4. We expected to have good results for low mobility (i.e.: small speeds and large pause times), where by good results we understand a high probability to have a large percentage of the network nodes in the largest spatio-temporal component. Such was the case indeed (note the case of speed = 0.5m/s and pause time = 50s) where about 90% of the nodes are located in the same component in about 90% of the cases. If we analyze the cases with varying pause times and constant speed = 10m/s, another interesting issue is that the worst results are obtained with a high pause time. It seems that in cases of higher node speeds, the spatio-temporal connectivity is better if nodes do not rest/pause. Such a result could be summarized in one phrase Move fast but do not rest.

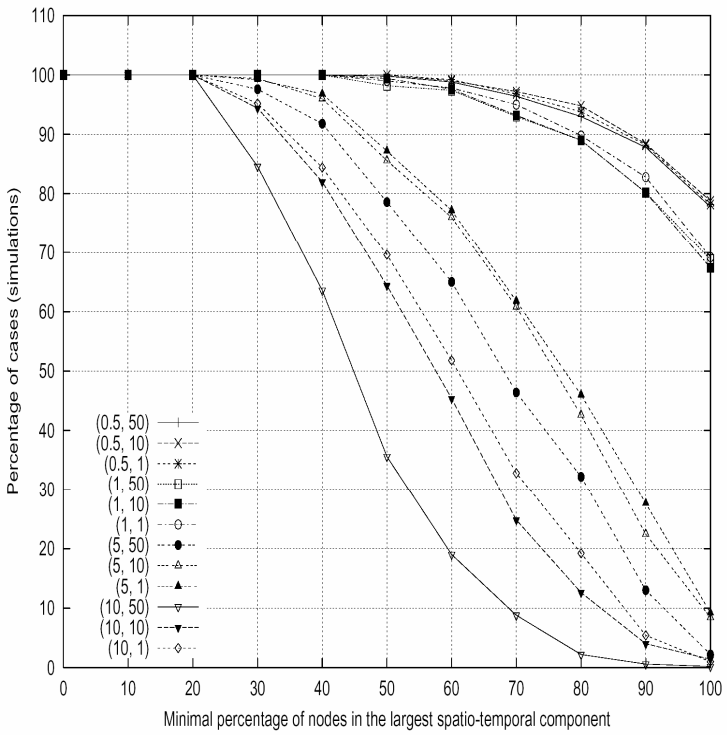


Figure 5-13 Impact of mobility on the management method

Conclusions and perspectives

We defined a new probabilistic scheme for configuring the management plane in an ad-hoc network. The underlying key idea is the notion of spatio-temporal connected nodes. A spatio-temporal connected component is a subset of the ad-hoc network, such that nodes within such a component have a high probability of being directly connected. The management plane is restricted to the largest spatio-temporal connected

components. We defined the underlying management self-organizing algorithm for both extracting spatio-temporal connected components. We also described in [52] several elective mechanisms to select the manager nodes in an efficient manner. The probabilistic approach was integrated into the ANMP management framework over a piggybacked ad-hoc routing protocol. The management approach is called probabilistic since its behaviour can only be guaranteed in a stochastic way. We can ensure that a given percentage of the overall network will be managed with a fixed apriori known probability. We have estimated these probabilities using extensive network simulations and we have performed a fine grained analysis of the mobility impact on our framework. Although network simulators present functional limits [60], the experimental methodology is generic and can be applied to other mobility scenarios. We are working on alternative mobility models including the Manhattan grid model and the reference point group mobility model.

5.2.3 Policy-based management of ubiquitous environments

There exists an apparent and increasing interest towards ubiquitous environments and ad hoc networking can be viewed as a real life realization of such environments. In ad hoc networks, mobile nodes are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Conventional wireless networks require some form of fixed network infrastructure and centralized administration for their operation. In contrast, since MANETs are self-creating, individual nodes are responsible for dynamically discovering other nodes they can communicate with. This way of dynamically creating a network often requires an equally dynamic ability to rapidly initialize, deploy and manage services and protocols in response to user demands, higher-level management goals and surrounding conditions.

We believe that this highly dynamic environment can benefit from the concept of Policy Based Network Management (PBNM) and the emerging context-driven autonomic communications trend. One of the major advantages of introducing controlled programmability to the system through policies is that it can offer an efficient and balanced solution between strict hard-wired management logic and unrestricted mobile code migration and deployment. There has been previous research on deploying PBNM solutions for MANETs, our work though introduces a novel organizational model specifically targeted to cater for the needs of MANETs, and ubiquitous networks in general, incorporating context awareness to dynamically adapt to the continuously changing conditions. Context-information can be used to trigger cross-layer changes (network and application configurations) according to network policies, leading to autonomic decision-making. Combining these two fields we can achieve autonomic communication solutions in MANETs, a very important aspect due to their inherent nature.

The proposed framework introduces a novel policy-based organizational model which combined with context information processing can effectively manage a MANET. Solutions to several issues are provided by the model's features such as its hybrid approach, the hyper-cluster formation, a distributed and replicated policy repository and the context-driven policy enforcement.

Distributed and hierarchical model

We adopt a hybrid approach by proposing a distributed and hierarchical model. Using the policy-based paradigm and context awareness we aim to provide a suitable

organizational model for ubiquitous networks. We apply our ideas in the context of Mobile Ad Hoc Networks (MANETs).

Before analyzing the proposed model we first explain the differentiation between node “modules” and “roles” (Figure 5-5). The three “roles” refer to the MN (Manager Node), CH (Cluster Head) and CN (Cluster Node) roles, as used in clustering schemes. Beyond the traditional duties for these roles, their behaviour and mission is guided by the defined policies. A “module” is the preinstalled software of a node. Our design has two modules: CM (Cluster Manager) and TN (Terminal Node).

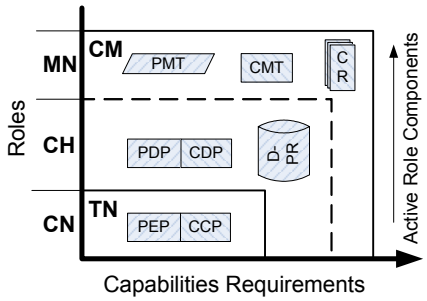


Figure 5-14. Node roles and modules

This module separation was deemed necessary to accommodate a wider range of node capabilities in the MANET. The TN is the simplest module and is lightweight to make it suitable for limited capabilities devices, e.g. smart mobile phones. Thus TNs can only be assigned to the CN role. On the other hand CM modules have full PBNM system functionality and context processing capability. Therefore CMs are collaboratively responsible for the network management and can be assigned to all three roles.

The selection of the appropriate module for each network device mainly depends on its capabilities. A set of minimum requirements offers a prescribed guideline and indicates whether a device can efficiently host the CM module. The set depends on the actual module implementation and refers to the capabilities to host a Directory Server (within DPR) and support XML processing (within CMT and CR). Effectively the differentiation between “roles” and “modules” refers to the differentiation of the organizational role of an entity in the network as opposed to the actual software capabilities it carries. The node roles and modules are depicted in Figure 5-5 where their respective policy and context related components are also shown. Depending on the assigned role of a CM, the respective components are either active or dormant. During initial network setup, default policies stored in CM are loaded for each role in order to guide the model’s deployment.

Based on the above classification, we present our policy-based organizational model. The multi-manager paradigm and the hyper-cluster formation are introduced, aiming to offer a balance between the strictness of hierarchical models and the disorder of distributed ones. At the same time our model embraces both as it can be deployed as either of these. In addition, policy-based management offers controlled programmability to the system to suit the MANET dynamic environment. The components for each of the three roles are depicted in Figure 5-6 as well as the information flows between them.

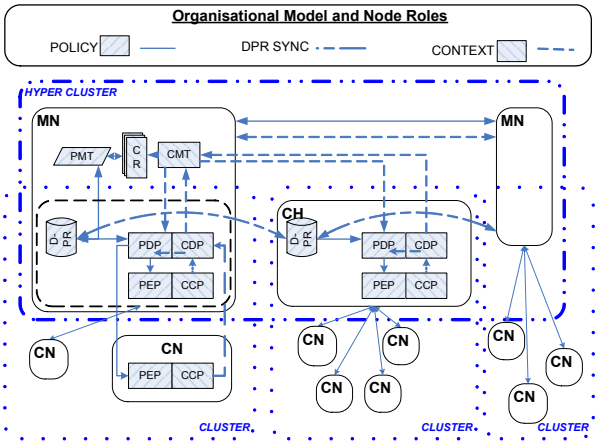


Figure 5-15. Organizational model

The idea behind the multi-manager paradigm lays in the nature of ad hoc networks and the purpose of their formation. Having more than one manager gives the flexibility to form networks between distinct trusted administrative authorities. This is performed without any of these being forced to forfeit its management privileges. Instead managers cooperatively introduce policies which guide the overall network's behaviour. For example, a MANET is setup for a corporate meeting between two companies' representatives. Multi-manager paradigm treats the companies' managers as equals and allows both to affect network behaviour by introducing policies. In addition, from a functional point of view, in large scale ad hoc networks scalability issues demand more than one manager in order to control and administer effectively the numerous cluster heads. A deployment example is depicted in Figure 5-7, mirroring the organizational model in Figure 5-15.

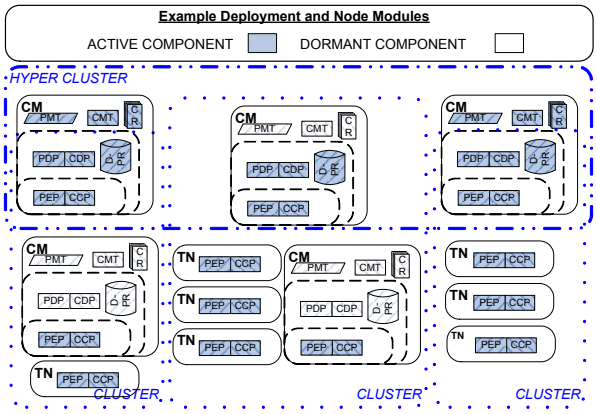


Figure 5-16. Example deployment

Hyper-cluster formation

We introduce the hyper-cluster notion referring to a set of Cluster Manager (CM) nodes that are assigned the MN or CH roles, utilizing available context information. The construction of the set is discussed in detail later. Nodes that hold the MN role encapsulate the CH role as well. The assignment of a node to the MN role depends on the MANET formation purpose and its application use. In the corporate meeting example the two devices which the companies' managers use are assigned the MN roles. In a military oriented scenario, e.g. platoon leaders would become MNs. If there is no apparent specification for the MN assignment or in dynamic conditions where MN re-assignment is necessary, then this occurs in an algorithmic fashion as described later.

Future work will consider the utilization of context information or a reputation-based election procedure for the selection of MNs.

Of fundamental importance to our architecture is the Capability Function (CF) of a node. It denotes its current ability to host resource-consuming software modules. The CF reflects two aspects of the nodes' capabilities, one referring to their computing attributes and another to their mobility (Mobility Ratio - MR). The driving concept is that if a node moves constantly and is responsible for link breaks, then it should not be deemed as capable. In our approach the CF considers specific computing attributes, namely memory (MEM), processing power (PP), battery power (BP) and computing load (CL). MEM, PP and BP have a proportional relationship with the CF and MR and CL an inversely proportional one. By assigning weights to these variables based on their significance and incorporating the time aspect we deduce Equation 1.

$$CF(t) = \frac{w_1 \times MEM(t) + w_2 \times PP(t) + w_3 \times BP(t)}{w_4 \times MR(t) + w_5 \times CL(t)} \quad (1)$$

Effectively, the CHs together with the MNs form the hyper-cluster and collectively manage the MANET. Every node with CM module is registered to it and can access it in one hop. Every CN registers itself to its CH neighbour with the highest CF value, while those that do not have such neighbours acquire a route to one of them through their CN neighbours. Depending on the application use of the MANET, the MNs are either dynamically introduced in the MANET or statically configured upon the initial construction of the MANET. In the latter case our algorithm takes this into account by assigning explicitly these nodes to the MN role and thus the hyper-cluster. In the first case the aforementioned algorithm is executed once again upon the selected set of CHs to deduce the set of MNs of the MANET. The result is a clustered MANET with nodes in all three of the defined roles. Space limitations avert us from further describing the algorithm used to maintain the hyper-cluster when node movements affect network topology.

Distributed & replicated PR

The policy repository (PR) is a critical component in every PBNM system and we cannot rely on a single node to store this repository. The idea of storage replication is not new and is widely used in fixed networks as a backup in case of a major failure. In ad hoc networks however due to the intermittent nature of wireless links, it is expected that nodes will become disconnected frequently. Thus access to a central repository cannot be guaranteed depending on the networks volatility and mobility. In order to tackle this deficiency we propose the DPR (Distributed Policy Repository) component.

DPR is an enhanced version of the Policy Repository and is consisted of repository replicas distributed among hyper-cluster's nodes. Instead of simply replicating the PR among the nodes, we incorporate a sophisticated policy-based replication scheme. By utilizing context information (i.e. the Mobility Ratio) and based on the introduced policies, the system automatically enforces the appropriate replication state among hyper-cluster nodes, depending on how volatile the MANET is. In this way, we provide alternative access options in case a repository is corrupted or disconnected and distribute traffic load and processing overhead among nodes.

In effect policies and context guide the DPR behaviour and replicas' distribution, ensuring on one hand maximum repository availability (distributed copies) and on the other hand a single logical view of the stored policies (replicated content). Thus, efficient

management of clusters can be achieved even when temporarily disconnected from the network manager.

Apart from the policy related components as these are identified by the IETF policy framework we introduce a group of new entities to our system related to context collection and processing. These are necessary for our system being capable of sensing, communicating with its surrounding environment and adapting to the changing conditions. Incorporating context awareness into our policy-based management framework constitutes this as extremely flexible and dynamic in response to the inherently unstable MANET domain, allowing for a degree of autonomy to be reached.

Policy-based Management

In order to apply the PBNM paradigm to our system we adopt the standardized by IETF/DMTF information model for policy representation. Defined policies are represented according to PCIM/PCIMe [66], [68]. Based on this representation we map policies to the standardized LDAP data model [69-71].

Our initial efforts focus on the definition of the necessary policies for MANET management, rather than the formal definition of a policy language. We believe this representation is both effective and lightweight so as to cater for the policy needs in the resource-poor MANET environment:

```
{Roles} [TimePeriod] if {conditions} then {actions}
```

We define the policy's "enforcement scope" as the set of nodes where actions need to be enforced, when the policy is triggered by this set's collected context. Regarding the actual policy design, we model realistic example policy types in order to illustrate our concepts. These policies are a first step towards a flexible and adaptable management framework specifically designed for the needs of a MANET. Based on the above definition, three enforcement scopes are realized for the needs of our design:

- **Network wide - Routing adaptation**

Policies can be triggered at the Manager Nodes by the context collected and aggregated from all network nodes. MN's PDP decide and enforce the actions network wide to all MANET nodes' PEPs. These policies are identified by their assignment to the MN node role.

Routing adaptation example policy: a plethora of protocols has been proposed to solve the multihop routing problem in MANETs, each based on different assumptions. A generic classification can distinguish them into proactive and reactive, in regard to the strategy used to establish routes between nodes. Performance measurements show that in high mobility networks where link breaks are very frequent, a reactive protocol (e.g. AODV) performs better since route establishment is dynamic and on demand. On the contrary, proactive protocols (e.g. OLSR) establish routing tables and perform better in relatively static MANETs.

Based on the above, we decided to model a policy which would enable dynamic on the fly adaptation of the routing protocol. Network conditions and context on one hand and administrator defined management goals on the other, can both be expressed by this type of policy which effectively alters routing strategy and increases network performance:

```
{MN}[T] if {RM=(n..m)} then {RoutProt:=k}
```

This policy type is used to adapt network behaviour by switching the routing protocol (RoutProt) according to the network's relative mobility (RM). RM is aggregated context information extracted from the network-wide knowledge of node movements, GPS positioning data, mobility ratio and other context. The simple condition monitors if RM value lies within the range (n..m) in order to enforce the associated simple action that activates the appropriate routing protocol. In our implementation k ={1,2}, where 1=OLSR and 2=AODV. The idea is to use a proactive routing protocol (OLSR) when relative mobility is low and a reactive (AODV) when high. Depending on management goals and on network-wide aggregated context, compound conditions and actions can be introduced in order to take in mind more parameters.

The network-wide enforcement scope of this policy implies that the condition variables used should have an aggregated network-wide value. The RM value for example is extracted from the aggregation and processing of node context like Mobility Ratio (MR) and it is collected at the Context Management Tool (CMT) components of the Manager Nodes. This higher level context information will drive the triggering of actions that should be enforced globally. Each CMT forwards this value to the local PDP and to the PDPs of all CH's under their MN. Each PDP enforces the triggered action to all cluster nodes and locally. Finally all CHs report successful execution to their MNs.

Hyper-cluster wide - Repository replication

Policies can be triggered at all hyper-cluster nodes by the context aggregated within the hyper-cluster. Decisions are enforced only at the hyper-cluster nodes. These policies are identified by their assignment to MN and CH node roles only.

Repository replication example policy: The need for repository replication has already been detailed. For this purpose we model a policy type to guide the replication degree of the Distributed Policy Repository. A manager node has the ability to dynamically define the behaviour and the replication degree of the DPR by introducing related policies on the fly and without shutting down the system or the DPR component.

```
{MN,CH}[T] if {FM=(n..m)} then {ReplDegState:=k}
```

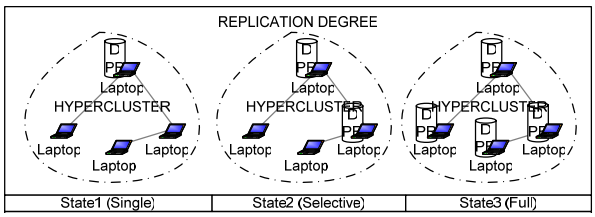


Figure 5-17. Replication Degree States

This policy type is used to guide the replication degree (ReplDegState) of the Distributed PR (DPR) component. The fluidity metric (FM) is a cluster-wide aggregated value similar to the RM metric. We implement three states of replication, namely k=1:Single, k=2:Selective and k=3 Full (Figure 5-17). These states reflect the need for PR replicas within the hyper-cluster nodes and adapt according to the volatility of the MANET. As mentioned earlier, the idea is to increase the DPR replication degree when network fluidity increases.

Based on cluster-wide information, such as the FM, the cluster's CDP informs the collocated PDP and policies of this type may be triggered for hyper-cluster wide enforcement. Each action {*ReplDegState:=k}* implies a state transition to State k and is enforced differently. A transition to full replication (State 3) is succeeded by activating

the DPR components at all nodes participating in the hyper-cluster. MNs acquire a master copy first, which the underlying CHs automatically replicate to their DPR. In order to transit to selective replication (State 2) a selection algorithm, similar to the roles' selection one is executed, this time only among the hyper-cluster nodes. A dominating set is selected to host the active DPR components. MNs are guaranteed to have an active DPR while the remaining hyper-cluster nodes have a list with their DPR-active neighbours. Finally the transition to single replication is achieved by keeping active the DPR of the MN with the highest capability function (CF) and maintaining backup files at the nodes with the next highest CF value. So, policy instances of this type implement the adaptation of the DPR replication degree according to network fluidity changes.

Cluster wide - Energy conservation

Policies can be triggered at all hyper-cluster nodes by the context aggregated within their cluster. Decisions are enforced only at the cluster nodes that triggered the policy. These policies are identified by their assignment to all three roles (MN, CH, CN).

Energy conservation example policy: A major issue in MANETs is the conservation of their device resources. We tackle this by introducing a policy type that adaptively configures energy consumption according to their current state and environment as well as the overall management objectives:

```
{MN,CH,CN}[T] if {BP=(n..m)} then {TransPow:=k}
```

This policy type is used to manage effectively the device resources by influencing relevant configuration parameters. The Battery Power (BP) metric is used here to affect the node's transmission power (TransPow). In our implementation k ={1,2}, where 1=Normal Power and 2=Low Power. The idea is to use a threshold battery level in order to reduce transmission power and conserve remaining battery power. Depending on management goals and on the collected context, a compound condition and/or action can be introduced. Policies of this type only need cluster-wide context knowledge since their enforcement is independent among clusters. The PDP of every Cluster Head receives context information for the registered variables from its collocated CDP and enforces the actions to the PEP of the reporting C. In these cases context information is withheld within the cluster, thus reducing overall traffic load and processing resources.

Conclusions

We have presented a novel policy-based management framework, specifically targeted at mobile ad hoc networks (MANETs). We believe the ideas presented can be a precursor for a generic framework for the management of ubiquitous environments. Based on the introduced distributed and hierarchical organizational model we deploy a replicated Distributed Policy Repository (DPR) among the hyper-cluster's nodes. By exploiting context information we can achieve effective evaluation of policy conditions that trigger actions with specified enforcement scope.

5.3 Self-Configuration of Ubiquitous Environments

Ubiquitous systems are characterized by their heterogeneity and the diverse capabilities of the nodes forming the network. Almost any device equipped with a wireless network interface can join a ubiquitous system in an ad hoc manner. In such an environment it is difficult to deploy common services without a common understanding among the participating nodes and their capabilities, in terms of processing power, battery life and

expected residence time and also in terms of already installed and operational software. In this section we present two different approaches to achieve self-configuration. First we present an event based approach and secondly a middleware-based programmable infrastructure. The latter allows the nodes of a ubiquitous network to download and activate required protocol and service software dynamically. This enables the alignment of the nodes' capabilities so that common services and protocols are used among heterogeneous ad hoc network nodes. The proposed programmability approach is implemented using wireless nodes forming a Mobile Ad hoc Network (MANET). The MANET can be viewed as a real life implementation of a ubiquitous system and we use these two terms interchangeably.

The concept of mobile ad hoc network has recently received significant attention due to the increasing popularity of ubiquitous computing and the rapid growth of wireless networking. In ad hoc networks, the mobile nodes (MNs) are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Typically this kind of network operates in a standalone fashion or may be connected to an infrastructure-based network through a gateway, e.g. wireless LAN access point, base station, etc. Conventional wireless networks require as prerequisite some form of fixed network infrastructure and centralized administration for their operation. In contrast, since MANETs are self-creating, self-organizing and self-administrating, individual nodes of the network are responsible for dynamically discovering other nodes they can communicate with. This way of dynamically creating a network often requires the ability to rapidly create, deploy and manage services and protocols in response to user demands.

5.3.1 IC Event-based approach

The discovery service is a key element in self configuration. It is responsible for detecting new devices or other SMCs when they come into communication range. It is responsible for vetting a device for membership by obtaining the device's profile which describes its capabilities and includes authentication information which is passed to the security service. If a new device passes the vetting procedure, the discovery service generates a component-detected event. By making the discovery service policy driven, it can easily be adapted to different applications.

This service is also responsible for determining when a device permanently leaves the cell and for generating a corresponding component-left event. Mobile components such as autonomous vehicles may temporarily be out of communication range. The discovery service has to distinguish between permanent departures from the cell and temporary loss of communication. Thus the number of failed polls to detect a departure from the cell and the frequency of polling a component is dependent on the SMC application so must be configurable via policies. We have implemented a very simple discovery service which runs on the PDA and broadcasts its identity message (id; type[; extra]) at frequency ω_R. A new device responds to the router identity message with a unicast device identity message. The discovery service can then query the device to obtain a device profile describing the services; it performs some basic vetting of devices, informs the device whether it has been accepted for membership, and if so generates a component-detected event which results in the device being assigned to specific roles depending on its profile and possibly on the credentials it possesses as indicated in Figure 5-1.

Each member device unicasts its identity message to the discovery service at the frequency ω_D. If the discovery service misses n_D successive messages from a particular device, it concludes that the device has left the SMC permanently, and generates a corresponding component-left event. If the device misses n_R successive discovery service identity broadcasts, it declares that is no longer a member of that cell.

When a device joins an SMC, it will not respond to a discovery service broadcast from another SMC. In the healthcare scenario, a sensor should not decide that it has left the SMC because there is a problem with the discovery service and then join the SMC of the person sitting next to the patient on the bus. One approach is to use pairing buttons on the PDA and device which have to be pressed simultaneously, while the devices are in radio contact, in order to set up an association with a new SMC, as with simple Bluetooth devices but we are also investigating more secure mechanisms for authentication and exchange of encryption keys based on identity based encryption.

Event bus

We have chosen to implement the event bus as an at-most-once, persistent publish/subscribe delivery service, using a router to distribute events to subscribers. The router is content-based – i.e., a subscriber specifies a filter when it registers, and all

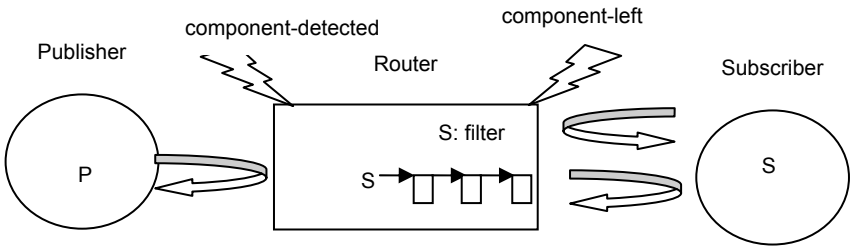


Figure 5-18 Publish Subscribe Event Bus

published events that match the filter are forwarded to that subscriber. The structure of the event bus is shown in Figure 5-18.

Publishers do not need to register with the Router. When a publisher sends an event to the Router, it does so synchronously and reliably; this reliable delivery is shown by the request/response arrow. Successful delivery of the event to the Router transfers responsibility for subsequent delivery to the Router. The Router attempts to deliver such an event to each subscriber whose filter matches the event. If it is unable to deliver the event to a particular subscriber due to transient communication failure, it queues the event for redelivery to that subscriber. The router attempts to deliver queued events until it knows that the subscriber is no longer a member of the SMC. Each subscriber is guaranteed to receive all events from a particular publisher in the same order as received by the Router. This is required in case there is a causal relationship between events from a particular publisher. If the Router receives a component-left event, it removes that subscriber's filters (if it had registered for any events), and purges any queued up events for that subscriber.

We do not assume that all communication within the SMC takes place via the event bus. Events are used to trigger policy actions. For example the policy service may use simple broadcast messages and unicast messages for polling individual components. Other components may use remote procedure calls or object invocations depending on what they support.

Scenario

We briefly describe a ubiquitous healthcare scenario to illustrate the concepts of the event driven self-configuration. Figure 5-10 shows the roles in a patient monitoring SMC with body sensors for monitoring, context, heart, temperature, blood-sugar levels etc. The medic role is for interacting with a medic attending the patient. Sensors typically report events when readings are beyond a threshold or can log actual reading if required, to the PDA, which can summon assistance via the phone if needed. The following section describes some example policies.

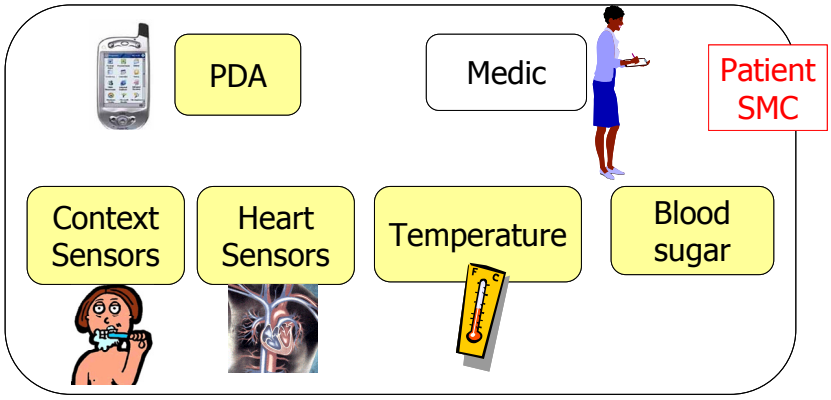


Figure 5-19 Patient SMC

Configuration Policies

We have had considerable experience of the use of policies as a means of specifying adaptive behaviour in network management and other applications. The use of separate, interpreted policies means they can be easily changed without shutting down or recoding components. Authorization policies should be enforced on the target components they are protecting as these must make the decision whether to permit or deny access.

Obligation policies are event-condition-action rules. For example the discovery service would have policies to assign a discovered nurse SMC with a public key certificate signed by the nursing council to the nurse role and a discovered Temperature sensor to the temperature role

1. oblig on newDeviceEvent (X, Xprofile) -> /roles/medicRole.add (X, Xprofile)

when Xprofile.type = nurseT & signed (Xprofile.certificate, nursingCouncil_PK)

2. oblig on newDeviceEvent (X, Xprofile) -> /roles/temperatureRole.add(X, Xprofile)

when Xprofile.type = tempSensorT

When a device is assigned to a role, the policies relevant to that role are loaded into the device. For example, the following policies corresponding to the temperature role would be loaded into an intelligent temperature sensor capable of interpreting policies:

3. oblig *every* (*mins* (2)) -> **raise local** tempEvent (*read_temperature*())

4. oblig on tempEvent (temperature) -> /pda.*reportTemp* (temperature)

when temperature > MAX_TEMPERATURE

Policy 3 tells the sensor to read the temperature and raise an event every 2 minutes and is triggered by a local time event. Policy 5 will report the temperature to the PDA

when the temperature value is greater than a maximum value. Wireless communication is only used if there is a problem with the temperature and, as a result, this minimizes power consumption.

Obligation policies can also be used to manage other policies in terms of selecting, enabling and disabling policies [56] to adapt overall behaviour of a SMC to current context e.g. a set of policies for when the patient is sleeping and a different set related to situations when the patient is doing normal activities. Policy 5 is triggered by the sleeping event from the context sensing service, and enables the policies for sleeping and disables those for the awake state, while Policy 6, triggered by the awake event, does the opposite.

5. **oblig on** sleepingEvent () -> /pda/policies/sleeping.*enable*(), /pda/policies/awake.*disable*()

6. **oblig on** awakeEvent () -> /pda/policies/awake.*enable*(), /pda/policies/sleeping.*disable*()

This obviously depends on the ability to detect current activity, which requires fusion of information from multiple sensors.

Authorization policies are needed to indicate what actions can be invoked or services accessed by a role. These should be enforced on the target components they are protecting as these must make the decision whether to permit or deny access. For example, policy 7 authorizes members of the temperature role to reportTemp on the PDA role.

7. **auth+** /sensors/temperature → /pda.*reportTemp*

Related Work

IBM has been the prime mover towards autonomic computing [60] and HP is also addressing similar issues in on-demand Utility Data Centres [61]. However most of the industrial work focuses on large clusters and web servers whereas we are concentrating on pervasive computing which is potentially more dynamic due to the mobility of components.

The Universal Plug and Play (UPnP) Architecture supports resource discovery and configuration of consumer devices (TV, video recorder, air conditioning etc.) which communicate via wireless within a home or office [58]. Although they concentrate on device configuration rather than configuration of software within nodes, some of the protocols and XML service specifications can be adapted for our purposes. UPnP currently focuses primarily on self-configuration and does not support the adaptability required for healing, optimizing or protecting.

There are many publish-subscribe event services such as Elvin [62], XML-blaster [63], and Sienna [64], which we have used in test systems; unfortunately, none of these routers are designed to run on small devices such as body sensor nodes (BSNs) and PDAs.

5.3.2 UniS Programmability approach

Ubiquitous systems are characterized by their heterogeneity and the diverse capabilities of the nodes forming the network. Almost any device equipped with a wireless network interface can join a ubiquitous system in an ad hoc manner. In such an environment it is difficult to deploy common services without a common understanding among the participating nodes and their capabilities, in terms of processing power, battery life, expected residence time and also in terms of already installed and operational software. In this section we present a middleware-based programmable infrastructure that allows

the nodes of a ubiquitous network to download and activate required protocol and service software dynamically. This enables the alignment of the nodes' capabilities so that common services and protocols are used among heterogeneous ad hoc network nodes. The proposed programmability approach is implemented using wireless nodes forming a Mobile Ad hoc Network (MANET). The MANET can be viewed as a real life implementation of a ubiquitous system and we use these two terms interchangeably.

The concept of mobile ad hoc networks (MANETs) has recently received significant attention due to the increasing popularity of ubiquitous computing and the rapid growth of wireless networking. In ad hoc networks, the mobile nodes are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Typically this kind of network operates in a standalone fashion or may be connected to an infrastructure-based network through a gateway, e.g. wireless LAN access point, base station, etc. Conventional wireless networks require as prerequisite some form of fixed network infrastructure and centralized administration for their operation. In contrast, since MANETs are self-creating, self-organizing and self-administrating, individual nodes of the network are responsible for dynamically discovering other nodes they can communicate with. This way of dynamically creating a network often requires the ability to rapidly create, deploy and manage services and protocols in response to user demands.

There has been no proper previous research on deploying application-level services or routing protocols dynamically in MANETs, but such aspect is important in ad hoc networks. For example, while routing and Quality of Service (QoS) conform to standardized frameworks and protocols in fixed IP networks, there are many potential solutions for ad hoc networks that depend also on the characteristics of the particular ad hoc network, e.g. topology volatility, characteristics of radio links, capabilities of nodes, etc. Given the multitude of potential solutions that may be environment-dependent, programmability is of paramount importance to allow mobile nodes to be enhanced on the fly with the required capabilities in the ad hoc environment. In addition, application servers may migrate to more powerful devices that have the required capabilities while less powerful devices may outsource computing tasks (cyber foraging). Programmability is possible through recent advances in distributed systems technologies and transportable "execute-anywhere" software. We present a novel programmable middleware that is capable to support cooperation, adaptability and alignment with respect to the required basic capabilities and additional services of the devices that form an ad hoc network, allowing a degree of self-management to be achieved.

A key aspect of MANETs, given their fluidity, is the possibility to adapt to their environment through context-awareness for many cooperation and coordination scenarios. The sheer amount of context information necessary for relevant adaptation places an important burden on the network, as potentially large amounts of data from diverse sources need be managed. This requires an infrastructure for sensing, collecting, and providing context information to applications [72][73]. The proposed programmable middleware takes this into consideration and takes an elementary step to provide such an infrastructure.

- **Applicability Scenarios**

The scenario presented above of routing protocol switching based on overall degree of mobility is a scenario we dealt with extensively as the first approach towards self-optimization in MANETs. We plan to study and experiment with other scenarios in the future as there are various possibilities to exploit context for MANET self-management.

A particularly scarce resource in MANETs is the battery power. In fact, if the battery is below a certain threshold, relevant nodes should still be able to send and receive packets but not to be used to relay packets if possible. In this case the high-level policy rule is *"a node should not forward packets if the energy level is below X%"* and the context information communicated is the remaining energy level of MANET nodes. In this case, this information could be exploited by power-aware routing protocols that will be configured to avoid using particular nodes as relays. Another aspect we are particularly interested is the identification of main streams of information in the MANET from and to particular nodes. Again this information could be used for alternative routing by QoS-aware routing protocols so as to keep the network load-balanced (self-optimization). Another use of this information would be to try and identify and subsequently isolate malicious intruders that send bogus data streams in denial-of-service attacks (self-protection). Context may also be gathered regarding access to important servers and these might be relocated or replicated if possible in order to provide a good level of service. Finally, many other uses of context information may be eventually possible. These management tasks are essential in order to establish efficient and reliable communication in Ubiquitous environments.

- **Programmable Ad Hoc Middleware Functionality and Architecture**

Our programmable platform follows a lightweight approach to achieve programmability through the use of loadable plugins. The latter are blocks of code that can be uploaded and executed on the ad hoc network nodes i.e. terminodes, in order to perform specific operations. This can be an installation of a new routing protocol, extensions to an existing one or any other function that the MANET could benefit from. In order to decide on a particular plugin to be used for a given scenario, a plugin election should take place utilizing the current contextual information. This involves advertisements from every node that can provide suitable plugins for a given election. A predefined election algorithm should be used and identify a plugin to install globally. The latter should then be distributed throughout the MANET in a peer-to-peer fashion. Following the installation of the elected plugin, the mobile nodes should activate, i.e. execute, the plugin. In the following sub-sections we describe the functionality and architecture of the middleware platform, highlighting important design decisions and presenting the relevant reasoning.

- **Middleware Communication and Components**

We have chosen to use the lightweight XML-RPC [74] protocol as the basis of communication between terminodes running the programmable platform. XML-RPC can be considered as a subset of the SOAP protocol, unburdened from unnecessary complexity. Like all remote call approaches, it allows software running on different operating systems and hardware architectures to communicate through remote procedure calls (RPCs). XML-RPC uses the HTTP protocol as transport and XML encodings for the RPC protocol itself. We chose an XML-based approach because we also use XML to represent contextual data in terminodes and this achieves easy integration. We could have possibly chosen Web Services, but this approach would have certainly been more heavyweight. In addition, Web Services, in the same fashion with distributed object technologies such as CORBA, necessitate object advertisement and discovery functionality, which is not required in our platform that relies on simple message passing, modelled through RPCs. Given our recent performance evaluation of XML and other management approaches [65], we believe that XML-RPC provides a useful blend of functionality and performance.

In addition, in order to make the platform even less demanding on processing power and portable to mid-range and small devices, we used the Java 2 Micro Edition (J2ME) virtual machine. The latter requires a much smaller memory footprint than the standard or enterprise edition, but at the same time it is optimized for the processing power and I/O capabilities of specific categories of devices. We used the Connected Device Configuration (CDC) framework instead of the limited (CLDC) one, as the latter lacks the required support of advanced operations.

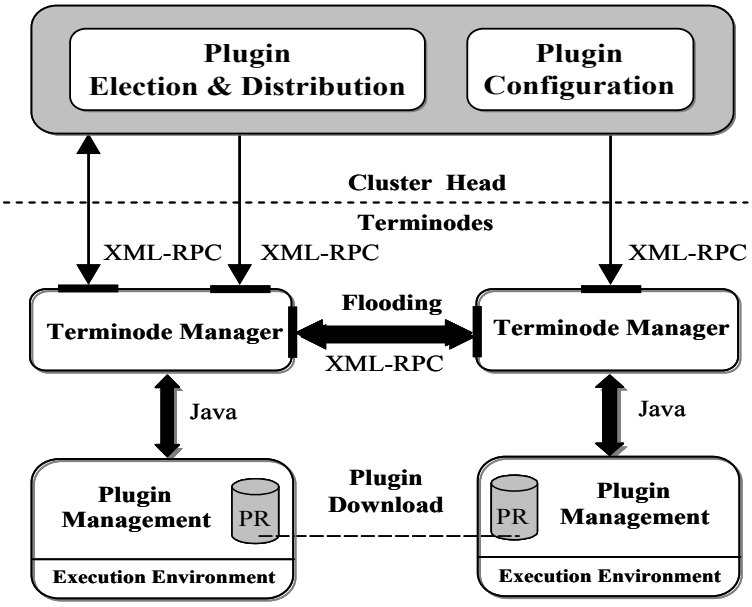


Figure 5-20 Middleware Platform Architecture

The programmable platform can be divided into two middleware modules according to relevant functionality, as shown in Figure 5-11: the Cluster (or Network) Head and the Terminode modules. Depending on a node's current status, middleware functionality switches between Cluster/Network Head and Terminode mode respectively. For the time being, we have implemented and validated Cluster Head but not Network Head functionality, since our experiments were conducted in a small testbed with 4 terminodes – we plan to implement cluster head cooperation and network head functionality in the future.

The Cluster Head components deal with the management of the programmable Terminodes regarding programmable plugin election and distribution, as well as (re)configuration. The key components of the Cluster Head module are:

- Plugin Election and Distribution: it is responsible for the initiation and coordination of the election process. It implements the plugin election algorithm and is responsible for the initiation, distribution and activation of the elected plugins across the ad hoc network.
- Plugin Configuration: it is responsible for the (re-)configuration of installed plugins across the ad hoc network. It is also able to modify on-the-fly key parameters of active plugins, if the latter support this functionality.

The Terminode components deal with the management of the programmable plugins regarding plugin advertisement, installation, storage, distribution and configuration. The key components of the Terminode module are:

- Plugin Management: is responsible for advertising available plugins to the Plugin Election component, listening for requests from the latter for distributing a specific plugin, communicating with its peer nodes for the purpose of exchanging network plugins, as well as for installing, executing, reconfiguring and terminating the operation of a particular plugin.
- Plugin Repository (PR): is responsible for storing and exposing available plugins to the Plugin Management component when required. The latter should be able to extract and advertise to the Plugin Election component the characteristics of the stored plugins for the purpose of plugin election. The Plugin Management component offers all the necessary operations for storing and deleting plugins, as well as searching for a plugin by name or type.

- **System Operation**

We describe here the complete system operation, from election triggering to plugin activation, considering an ad hoc network that consists of a single cluster. According to the current context or human user command, the CH triggers the election process by contacting Plugin Election and providing the type of the plugin currently required by the MANET. The Plugin Election object contacts all the terminodes of that cluster in order to request advertisements of candidate plugins. Each member node performs a lookup in its plugin repository for one or more suitable plugins that can satisfy the requirements of the election process and replies to the CH Plugin Election with the characteristics of the retrieved plugins. The CH Plugin Election executes the election algorithm and decides on the most suitable plugin, based on the current context, e.g. by assigning different weights to criteria such as CPU time, memory required, etc. Following the actual plugin election, the CH contacts terminodes that already possess the elected plugin and instructs them to distribute it across the cluster. Plugin distribution is carried out in a peer-to-peer fashion, with the “owning” nodes flooding the plugin to their peers, and so on. A key aspect of plugin flooding is that a terminode should not receive the same plugin twice. This is deliberately prevented as the plugin size might be considerable and prove costly to the network. In order to prevent this from happening, the transmitting node will have to first probe its peer if it has already acquired the plugin and then only transmit it. The plugin distribution includes apart from the actual plugin transfer, the transmission of the plugin characteristics. For the actual transmission, the terminode currently distributing the plugin contacts the neighbour node and passes all the characteristics of the elected plugin, followed by the actual transfer of the plugin’s execution code. When a new plugin has been successfully installed, the node sends a notification to the Plugin Election of the CH. At this point, the plugin is installed and available to be activated. The CH Plugin Election object, after receiving installation notifications from all the member nodes or after a predefined timeout period, it floods an activation message across the cluster to instruct the member nodes to execute the elected plugin. Each member node should then perform a lookup in its Repository for the plugin, in order to obtain its reference in the local file system and consequently execute it in user space.

5.4 Collaboration and detection of faulty nodes

We addressed in [75] the issue of detecting faults in ubiquitous/ad-hoc networks. Faulty behaviour and intermittence are closely related in ad-hoc networks: a node can have a regular intermittence due to mobility and other ad-hoc specifics, while faulty behaviour can generate abnormal intermittence behaviour. The key issue is to

differentiate abnormal intermittence from regular intermittence and thus identify faulty nodes from regular non-faulty ones. While fault detection in fixed wired networks is not hindered by the impossibility to observe a given node, ad-hoc networks specifics do provide major challenges with respect to this issue. A node that does not reply to legitimate polling in a fixed network is typically considered as not functional. In ad-hoc networks, observability is a major issue: a node might not be reachable because it is moving and is out of reach, or because it is not functioning properly.

We first introduce the concept of our detection scheme and briefly overview the routing protocol which serves as an underlying data source. We then define an information theoretic measure for monitoring node intermittence and several collaborative methods of abnormal intermittence detection. These methods are evaluated through an extensive set of simulations.

- ***Concept***

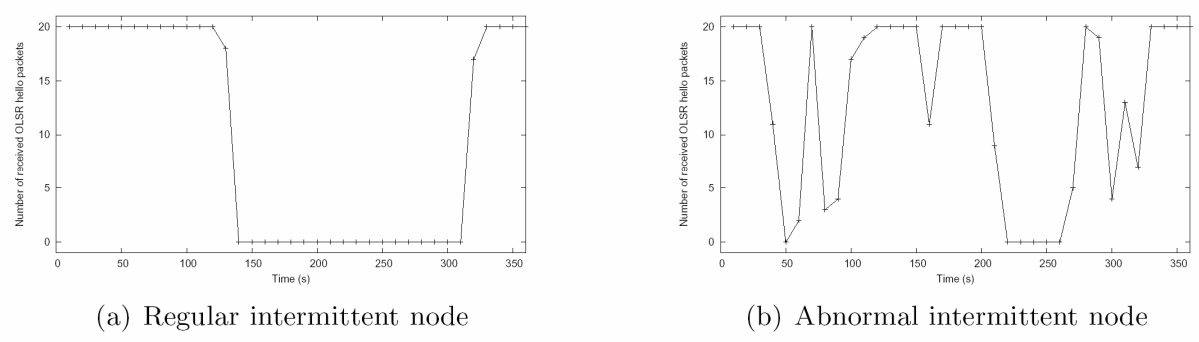
We propose to monitor the intermittence of ad-hoc nodes in order to detect fault nodes. Intermittence in ad-hoc networks is a relatively normal condition due to causes that are inherent to such a network: nodes are moving, connectivity might be lost for longer or shorter time-periods and battery life is a well-known issue for this target domain. However, intermittence might also have a different cause related to abnormal ad-hoc behaviour, where:

- Failures due to misconfiguration and errors at the physical layer might generate an atypical behaviour, where nodes will appear intermittent although from a mobility point of view they did not change significantly,
- Routing failures can be encountered when the routing process is affected by voluntary activity [76], malicious activity (attacks against the routing plane), errors in its configuration or at the protocol stack level,
- Abnormal mobility. While normal mobility is difficult to define, in some specific target deployment (for instance military applications), unpredicted mobility patterns can seriously impact the network resilience and service level.

We analyze the behaviour of intermittent ad-hoc nodes and propose an entropy-based approach for monitoring the routing plane and detecting fault nodes. We propose to monitor the optimized link state routing protocol (OLSR) by analyzing the distribution of hello packets received by each node during the beaconing operation. This is done in order to detect abnormal intermittent ad-hoc nodes. We assume a mobile ad-hoc network as a set of n mobile nodes $V = \{v_1, v_2, ..., v_n\}$ moving in a given surface during a time period T. The time period T is split in k measurement interval $[t_l, t_{l+1}]$ with $t_l = l \times T$ for an integer l in [0, k]. During the OLSR beaconing, each node v_i in V can receive hello packets from the other network nodes located at one hop. The number of beaconing hello packets received by a node v_i from a node v_j is noted Xv_i,v_j and can be considered as a random variable $Xv_i,v_j (l) : [0, k] \rightarrow [0, b_{max}]$ with l characterizing the interval $[t_l, t_{l+1}]$ and b_{max} the maximal number of hello packets that v_i can receive from v_j. If the hello packets emission interval r is supposed to be homogeneous among network nodes, then Xv_i,v_j is bounded by $b_{max} = 1/r \times T/k$ .This is not a limiting constraint, since the monitoring process can be easily extended to different (per node) r values.

- ***Intermittence Measure***

Since monitoring is performed at the routing level, we intent to detect abnormal intermittence due to multiple failure causes such as routing failures, battery problems, physical perturbations and pathological mobility. This monitoring is based on the analysis of how an intermittent node is perceived by a neighbour node or by a set of neighbouring nodes. An intuitive idea of intermittence perception by a node can be given by analyzing Xv_i,v_j values for a network node v_i for different v_j network nodes. Figures 5-12(a) and 5-12(b) depict these values respectively for a regular intermittent node and for an abnormal intermittent node. Each Figure represents the number of received hello packets Xv_i,v_j measured (on the y axis) for each time interval $[t_l, t_{l+1}]$ (on the x axis). In Figure 5-12(a), the regular ad-hoc node either generates a short distribution with most of the values equal to 0 when the node is not in the neighbourhood, and equals to bmax when the node is located in the same neighbourhood. In Figure 5-12(b), the abnormal intermittent node (as seen by the other nodes) is characterized by a larger distribution of Xv_i,v_j values.



(a) Regular intermittent node (b) Abnormal intermittent node

Figure 5-21 Illustrative example of the number Xvi,vj of hello packets periodically received by an ad-hoc node

Formal model

The main issue that we address is stated in two simple questions. Can we detect abnormal intermittence by monitoring simple parameters like for instance route state related ones? Can we do it in a distributed way such that malicious or non-cooperative nodes are out-weighted? The perception of an abnormal intermittent node by an observing node can be modelled as a discrete Markov chain with four states: these four states depend on the functional state of the observed node (node up or node down), but also on the location of this node compared to the observing node (1-hop neighbour or not).

From the perspectives of the abnormal intermittent node itself, the node behaviour can be reduced to a discrete Markov chain with two states {NODE UP, NODE DOWN} with the transition probabilities p-failure and q-recovery that a node goes down and respectively goes up after a failure. The stationarity equation can be resolved to get the unique stationary distribution of this irreducible and positive recurrent Markov chain, as presented in equation 1 where p_{up} is the probability to be in state NODE UP and p_{down} is respectively the probability to be in state NODE DOWN.

$$(p_{up}, p_{down}) = (\frac{q}{p+q}, \frac{p}{p+q}) \quad (1)$$

In order to evaluate the impact of node abnormal intermittence (parameters p and q) on Xv_i,v_j distribution, we consider a simple scenario where v_i and v_j are in the same neighbourhood, with v_i the observing node and vj the observed abnormal intermittent node. During the measure interval $[t_l, t_{l+1}]$ (presented in Figure 5-13), the probability of v_j to emit a hello packet at each r hello emission interval is given by p_{up}, the probability that the OLSR node v_j is up. Therefore, the probability for v_i of receiving k hello packets (and then the probability of not receiving (bmax − k) hello packets) during $[t_l, t_{l+1}]$ follows a binomial distribution presented in equation 2.

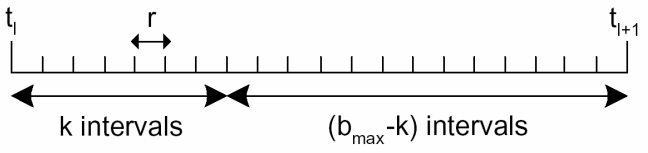


Figure 5-22 Measure interval [tl, tl+1] divided in bmax hello emission intervals

$$P(X_{v_i,v_j} = k) = \binom{k}{b_{max}} p_{up}^{k}(1 - p_{up})^{b_{max}-k} \quad (2)$$

This probability distribution will be considered to determine the impact of transition probabilities p and q on the observed fault behaviour.

Beaconing entropy

We can monitor the OLSR routing protocol by performing an entropy measure of the probability distribution of Xv_i,v_j. The entropy, defined by Shannon in [77], provides a measure of disorder for a system, where higher values indicate more disordered systems.

In our case, it characterizes the distribution disorder (largest distribution) of hello packets for a neighbour node. Equation 3 defines the entropy measure noted H (Xv_i,v_j) in a formal manner.

$$H(X_{v_i,v_j}) = \sum_{k=0}^{b_{max}} P(X_{v_i,v_j} = k).log(\frac{1}{P(X_{v_i,v_j} = k)}) \quad (3)$$

Let us consider the entropy measure for the examples presented in Figures 5-12(a) and 5-12(b). H(Xvi, vj) equals 1.307 for a regular intermittent node, while H(Xvi, vj) reaches 2.642 for an abnormal intermittent node. High values of H(Xvi, vj) identify a disordered distribution with values Xv_i,v_j largely covering the interval [0,bmax], and thus identify nodes with abnormal intermittence.

Assuming the discrete distribution of Xv_i,v_j given in equation 2, the entropy H (Xv_i,v_j) of this binomial distribution can be asymptotically approximated via analytic depoissonisation as proposed by Jacquet and Szpankowski in [78] (see equation 4 where a and k are explicitly computable constants).

$$\begin{aligned} H(X_{v_i,v_j}) &\asymp \frac{1}{2} ln(b_{max}) + ln\sqrt{2\pi p_{up}(1-p_{up})} \\ &\quad + \sum_{k\geq 1} a_k b_{max}^{-k} \\ &\asymp ln\sqrt{\frac{2\pi pq}{(p+q)^2}} + c \end{aligned} \qquad (4), (5)$$

In equation 5, the approximated entropy $H(Xv_i,v_j)$ is then given in function of the Markov chain's transition probabilities (p, q) (from equation 1) with the constant value c. The impact of parameters (p, q) in [0, 1] can be estimated by studying the partial derivatives of $H(Xv_i,v_j)$. Indeed, this probabilistic approximation of the entropy provides an approximation of the additional entropy generated by an abnormal intermittent node (additional to the one generated by the mobility model), perceived from the point of view of a local node. It shows how abnormal intermittent nodes can be detected by a local node, by selecting the network nodes with the highest entropy $H(Xv_i,v_j)$ of hello packets distribution. The reliability of this local measure can be improved by ad-hoc nodes collaboration.

- ***Collaborative detection scheme***

We presented in the previous section how the entropy measure of beaconing packets can locally detect abnormal intermittent nodes. The intermittence detection can be improved by sharing the local measurements among network nodes in a distributed manner. As depicted in Figure 5-14, each ad-hoc node v_1, v_2, v_4, v_5, v_6 monitors locally the network nodes and exchange their local measurements to detect the abnormal intermittent node v_8. We will detail several distributed methods to synthesize the local measurements and to provide a more efficient and reliable intermittence monitoring at the network scale.

A detection approach consists in (1) ranking the potential abnormal intermittent nodes in the ad-hoc network according to a criteria c and then (2) selecting abnormal intermittent nodes according to a threshold value λ (nodes selecting are those presenting a criteria value $c(v_j) > \lambda$). We propose three detection methods and describe them below:

- The first detection method m_1 (called majority voting) defines a ranking of potential abnormal intermittent nodes in function of the number of observing nodes (which perceived the node as abnormal intermittent) in the network.
- The second method m_2 (called entropy sum) takes into account the number of observing nodes, but also the entropy values measured by these nodes. Therefore, m_2 ranks potential abnormal intermittent nodes in function of the sum of entropy values in the network. This method is actually an adaptation of method m_1 where results are weighted by entropy values.
- The last method m_3 (called entropy average) ranks potential abnormal intermittent nodes based on the average of measured entropy values. m_3 does not focus on the number of observing nodes, but favours the entropy values at the network scale.

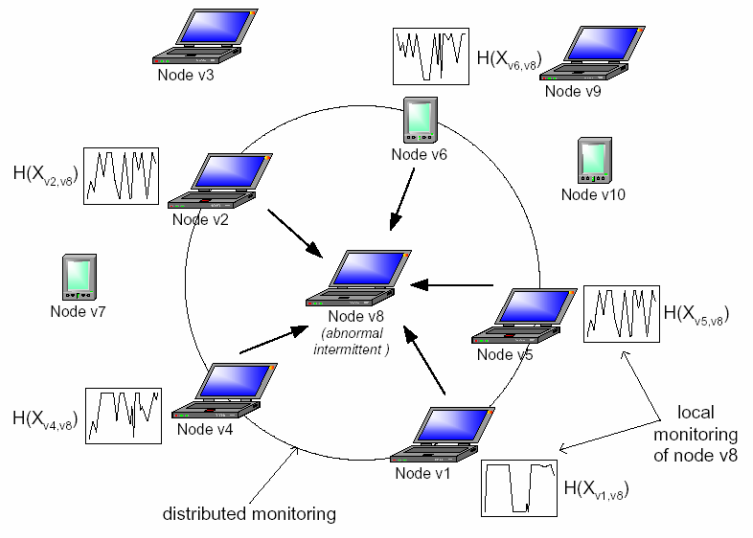


Figure 5-23 Collaborative detection of a fault node

These methods can be extended by weighting the measurements obtained from network nodes according to their reliability. Measurements from reliable nodes will have higher weights and then will be more taken into account in the detection process. The temporal coherence and the life time of monitoring data can be improved using approaches such as proposed in [79].

- ***Experimental results***

We performed a set of simulations in order to evaluate the performance of the entropy-based monitoring with the different proposed detection methods, and to estimate the impact of the mobility model on this approach. We simulated with ns-2 a mobile ad-hoc network of 50 nodes implementing the OLSR routing protocol and moving in a 1500 m x 300 m rectangular area during a time period of 900 simulated seconds. For each experiment, a set of abnormally intermittent nodes is randomly chosen and follows the two-state Markov chain model with transition probabilities (p, q). This set of abnormal intermittent nodes is then compared to the set of nodes detected as abnormal intermittent nodes by the detection scheme. In order to quantify the performance of the approach, we performed an analysis of sensitivity and specificity. Our approach can be seen as a diagnostic test, where we test if an ad-hoc node is abnormally intermittent (positive test) or regularly intermittent (negative test). We use the receiver operating characteristic (ROC) [80], a graphical plot of sensitivity (Sn) versus 1-specificity (1 - Sp), to evaluate the detection efficiency. The ideal diagnostic method shows a plot that is a point in the upper left corner of the ROC space, as sensitivity (all true positives are found) and specificity (no false positives are found) reach both 1.0. A diagnostic method becomes random (and then inefficient) when it presents a line at n 45 degree angle from bottom left to top right, because the number of true positives equals the number of false positives. In the next parts of this section, we will detail the experimental results (1) by plotting and analyzing the ROC curves to compare the performance of the three detection methods and (2) by evaluating the impact of mobility model (random waypoint model with parameters (pause, speed)).

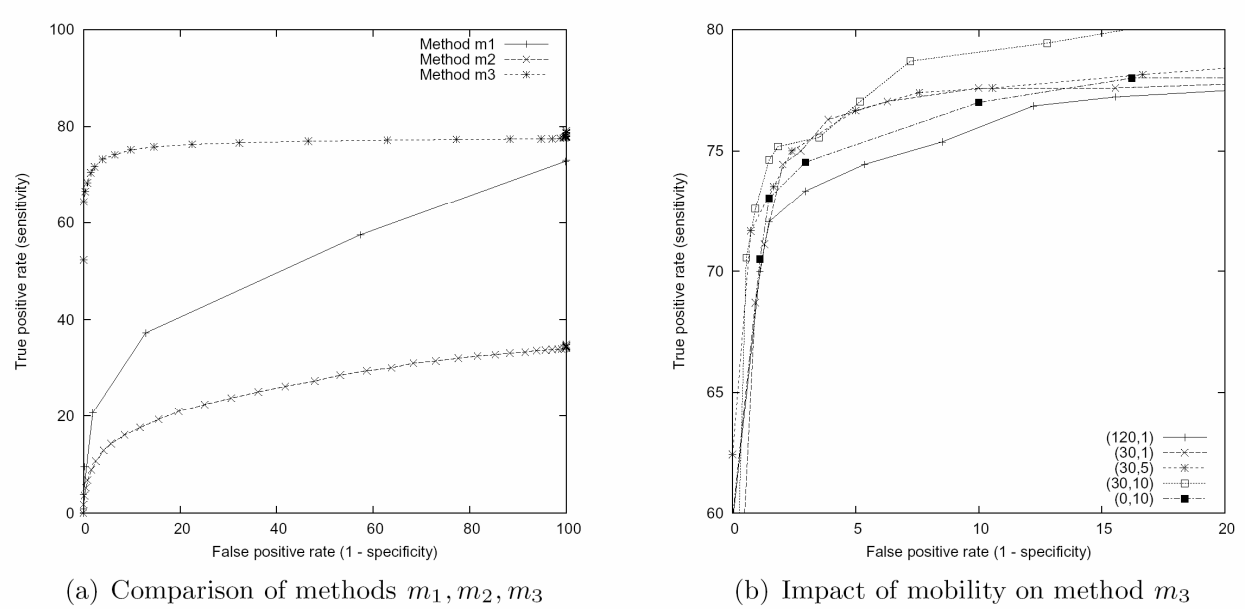
Performance of the collaborative detection methods

We first analyzed the performance of the three collaborative detection methods. These results are shown in Figure 5-15(a) and are based on an extensive set of simulations

with different mobility parameters (pause, speed) and abnormal intermittence parameters (p, q). We varied node mobility with pause time from 0 to 120 seconds and with speed from 0.1 to 10 m/s. The abnormal intermittent nodes were parameterized with realistic transition probabilities. The failure probability p was set with low values from 0.1 to 0.2 and the recovery probability q from 0.1 to 1.0. For each individual setting we performed 150 simulations to assure the non-bias of the result.

The performance of the detection methods is summarized on Figure 5-15 (a), where we plotted the ROC curve for each method. A point (x,y) on a curve stands for the true positive rate (y) of the method compared to the false positive rate (x) for a given threshold value. We are interested in an optimal diagnostic method providing a low false positive rate for a maximum true positive rate. The closer a method is localized in the upper left corner of the ROC space, the more it provides an efficient detection.

We can therefore deduct that method m_3 based on the average of entropies presents a better diagnostic test than the two others. In particular, method m_3 offers good results with a true positive rate of more than 70% in most of cases. In a more refined way, if we expect a false positive rate of less than 20%, method m_3 with 70% of true positive is definitively better than method m_1 providing a true positive rate of less than 45%, and still better than method m_2 showing a true positive rate of less than 20%. It turns out that methods m_1 and m_2 present less convincing performance, which can come from the simple fact that the detection is too dependent on the number of nodes observing an intermittent node. For instance for method m_1, the detection is based on the majority voting and consequently the probability of an ad-hoc node to be detected grows with the neighbour number of that node. In the same way, method m_2 considers the entropy sum at the network scale, which also rises in function of the number of neighbour nodes. In method m_3, using the average of entropies provides a more independent and reliable measurement of intermittence, where the increasing of the number of neighbours improves and refines the averaged measurement without denaturing it.



(a) Comparison of methods m_1, m_2, m_3 (b) Impact of mobility on method m_3

Figure 5-24 Evaluation of the collaborative detection methods with ROC curves

Impact of mobility model on intermittence detection

In a second series of experiments, we addressed the mobility issue, where different mobility parameters were evaluated with a realistic intermittence (parameters p = 0.1 and q = 0.4). We varied the random waypoint parameters with reasonable pause time from 0 to 120 seconds and speed from 1 to 10 m/s, and measured the sensibility and specificity of the entropy average method m_3. These results are presented in Figure 5-15 (b) where we plotted the ROC curves for each couple (pause, speed) of mobility parameters. We were interested in studying the detection method for configurations with low false positive rate and we therefore limited the plotting of ROC curves to a false positive rate no more than 20%.

The comparison of ROC curves shows that the impact of mobility is relatively limited for realistic mobility scenarios. The variation between the lowest and the highest mobility parameters is indeed less than 5%. This statement comes from the nature of our measure, which actually highlights more the additional entropy generated by abnormal intermittence than the entropy generated by the network mobility. We expected that higher mobility implies bad results (i.e.: high speeds and short pause times), where by a low result we understand a low true positive rate for a false positive rate of less than 20 %. Such was the case indeed (note the case of pause = 0 and speed = 50) where the sensibility is less than 72%.

A rather surprising result is however the case of lower mobility parameters where the sensibility is improved when mobility grows. This contradicts our initial hypothesis that mobility deteriorates our detection and leads us to more contrasted conclusion. The sensibility actually evolves in two steps. First, the detection is improved by mobility rise, from mobility parameters (120, 1) to (30, 10). The mobility increases the number of observing nodes per observed node. Second, the detection is noised with highest mobility scenarios and is not capable anymore to highlight efficiently abnormal intermittent nodes. In brief, the detection shows best results when mobility scenarios are not extreme (lowest and highest mobility parameters).

- ***Related work***

Among the pioneering approaches in our context of fault management (we do not focus on intrusion detection/security), Jakobson introduces in [81] an approach for correlating events and faults with temporal constraints. Failure detection algorithms based on keep-alive messages (active approach) are experimented in [82] and their performance are evaluated in overlay networks. The OLSR hello mechanism corresponds to one of the experimented keep-alive approaches called gossip approach where a node periodically sends ”I’m alive” messages to its neighbours. Related work in monitoring the routing plane for fixed networks is described in [83], where a real-time system tracks the routing state of a single OSPF domain, using flexibly OSPF snooping and link state SNMP tracking. This system offers network statistics based on the monitoring of a link-state routing protocol, but it is mainly designed for performance analysis rather than fault detection. The DAMON architecture [84] defines a distributed monitoring system based on agents for multi-hop networks: agents perform the network monitoring and send to data repositories the measurements data. DAMON supports multiple data repositories and includes an auto-discovery mechanism of data repositories by the agents. This generic architecture is not dedicated to specific network parameters and could therefore be appropriate for the storage of fault monitoring data. WANMon is a monitoring tool described in [85] to monitor the resource usage in terms of network traffic, energy,

memory and CPU, but its scope is limited to the host-level monitoring. Finally, our previous work in [86, 87] addresses an information model and a probe-based architecture for monitoring ad-hoc node participation.

- ***Conclusions and perspectives***

We proposed a lightweight and distributed fault monitoring approach for ad-hoc networks and addressed the issue of detecting abnormal intermittence of ad-hoc nodes. The proposed solution is based on two key concepts: (1) a measure based on information theory to monitor intermittence over the routing layer and (2) a distributed scheme to perform abnormal intermittence detection among the network nodes. We have shown how correlating monitored data from different ad-hoc hosts provides an efficient and reliable detection of abnormal intermittence. We have proposed and evaluated different distributed methods based on fault ranking and thresholding methods. The main advantages of our approach are multiple: we can detect abnormal intermittent nodes even if they are not instrumented. The monitoring process is passive and completely decoupled from the OLSR protocol, without requiring any additional routing protocol piggybacking. Our future work is consisting in defining an autoconfiguration mechanism for the collaborative detection methods and integrating the monitoring scheme into a management architecture.

5.5 Discussion

It has become obvious that the task of self-management of Ubiquitous environments is quite complex and several different approaches can be adopted. In order to successfully tackle this task, we propose some future directions which can be targeted by partner collaboration. Based on the presented research work in the area of autonomic management in ubiquitous networks several promising future research directions can be identified:

- The use of a Capability Function can be enhanced with additional metrics based on Information Theory calculations as well as probabilistic guarantees. The Intermittence metric can be included in the calculation of CF and the probabilistic approach can be used as a new heuristic in the algorithmic selection of the most capable nodes. Based on the work for ad hoc network management, we propose to integrate the probabilistic management approach with the policy-based framework. Their combination can provide a fruitful ground for collaboration and looks quite promising. The idea is to use probabilistic guarantees to assist in the selection of capable nodes to host the active Distributed Policy Repository components.
- In addition, the probabilistic management approach can be combined with the programmable middleware and the selection of the hyper-cluster nodes. The formation of the hyper-cluster can be achieved by creating a spatio-temporal component that has direct control of a guaranteed node percentage.
- In order to enhance performance of Ubiquitous environments the combination of active and passive abnormalities detections is also promising. The described use of Information Theory for passive detection of faulty nodes is naturally combined with the efforts in the area of malicious nodes detection. Both provide a promising collaboration field.

- ***Using fault detection information to organize the management plane***

LORIA/INRIA has proposed a distributed fault monitoring scheme to detect pathological nodes in an ad-hoc networks. An intermittence measure is introduced to analyze the routing traffic and infer-network faults. The solution is capable to detect pathological intermittence due to several causes such as battery failure, low system capacity, routing misconfiguration and physical errors. Several collaborative detection schemes are proposed to synthesize the monitoring data among the network nodes and to avoid biased local views. In the meantime, the University of Surrey has proposed to organize the management plane of their policy-based management framework using an election mechanism based on a capability function. In their approach, the capability function is used to detect if a node is capable to host and perform management operations. For instance, it takes into account computing attributes such as memory, processing power, battery power and computing load. It would be of major interest to introduce the intermittence measure of INRIA's fault monitoring scheme as a criteria of the capability function of Surrey's PBNM framework in order to discard nodes that show a pathological behaviour and elect the most capable.

- ***Designing a probabilistic programmable middleware***

The University of Surrey has developed a programmable middleware. This middleware provides support for aligning dynamically the capabilities of ad-hoc nodes based loadable plug-in election. The mobile nodes advertised different loadable plug-ins and then an election mechanism is responsible for selecting the set of plug-ins the network nodes have to load to establish a common basis of functions. LORIA/INRIA has proposed a probabilistic scheme for lessening the requirements on the management plane. It could be very interesting to optimize the programmable middleware by considering a probabilistic scheme. In particular, while the middleware solution uses the lightweight XML-RPC protocol as the basis of communication between mobile nodes running the programmable platform, a probabilistic scheme in that context could be further investigated to reduce the impact of the middleware on the overall network performance.

- ***Combining Misbehaviour Detection Schemes***

Both the University of Surrey and LORIA/INRIA Lorraine are working on misbehaving detection scheme. INRIA has proposed a distributed fault monitoring scheme: the intermittence of ad-hoc nodes is analyzed at the routing layer to detect pathological cases. The detection is performed in a distributed manner by synthesizing data information from the network nodes using three different threshold-based detection methods. They are currently investigating an auto-threshold mechanism to self-configure the detection methods. It would be a promising research direction to compare and combine the detection schemes of INRIA and of the University of Surrey.

6 Distributed Autonomics: Utilizing Distributed Management Patterns in Ensembles of Autonomous Devices with Cfengine

6.1 Introduction

The primary work in this part of the deliverable is to integrate the existing technology of cfengine [88], [89] with the distributed pattern algorithms developed by The Swedish Royal Institute of Technology (KTH) [90], [91].

Note: the use of the term policy is not specifically related to the notion of Event Condition Action (ECA) in this section of the document. Policy in this context means simply a set of pre-determined judgments about the freedoms, constraints and the behaviours of a device which apply at all times, not only when specific events arrive.

6.1.1 Autonomics: self-regulating systems

Cfengine is an established framework for autonomic computing, used on up to a million computers around the world. Management patterns are an efficient way of signalling in overlay networks enabling distributed collaboration. The aim of this part of the WP9 work is to allow cfengine to benefit from management pattern technology without sacrificing its basic principles of autonomy.

Autonomy and autonomic are two different concepts. Autonomy implies independence of decision-making. There is no single definition of what “autonomic” means, but it implies some kind of “self-regulation”. Cfengine’s behaviour is autonomic in the following senses.

- Each component device that runs cfengine is responsible for the operation of that device in relation to a given context or environment.
- Using only voluntary cooperation, individual devices can harmonize their behaviour and policies by reacting to environmental state or to one another.
- The state of each individual component is monitored, and this provides feedback within the component. Communication between components can be established on a voluntary basis but is never mandatory.

Cfengine runs on each individual device in a network and makes each device administratively self-reliant. If a device wishes to subordinate itself to the instructions of an “authority” then it must make this choice voluntarily. No external agent can force a device to comply with its wishes involuntarily.

Although some systems with a centralized management claim to be autonomic, it is seen as an important feature in this work to preserve the decentralized aspect of the components, while using management patterns to most efficiently communicate with neighbouring components.

6.1.2 Self-management for ubiquitous environments

In the context of EMANICS, *ubiquitous* means wireless and roaming. Cfengine can be run on any wireless device with a suitably Unix-like operating system, and it is widely used to manage Linux laptop computers. The level of management between devices can vary enormously, from a data-centre multi-processor to a hand-held PDA. The amount of management employed is entirely a matter of context and policy.

Cfengine makes no special exceptions for wireless computing; rather it supports the idea of continued operation under *partial connectivity*. There is no essential difference between fixed and wireless communication [92]. Cfengine uses network communication opportunistically rather than as an essential dependency. When the network is unavailable, local regulation tasks can be performed without the need for interacting with neighbouring devices.

The description of partially reliable networks as ad hoc networks has been given in ref. [92]. This work was follows up within the EMANICS network of excellence by INRIA. Cfengine implements probabilistic management in the sense of Badonnel et al [93], [94]. Cfengine "strategies" can be used to weight distributions of policies based on the state of a device. Moreover, the agents record the average connectivity of its neighbours during attempted communication and can use this information to perform inter-peer "neighbourhood watch" monitoring of one another.

6.2 Introduction to Cfengine

The Cfengine project was started in 1993 by Mark Burgess in Oslo and is one of the first and longest running autonomic computing projects, focusing originally on distributed configuration management and later on resource regulation [89], [88]. Cfengine is widely used, installed on an estimated million Unix and Windows computers all over the world. It runs on everything from small laptops (or Linux watches) to large Cray machines.

Cfengine, or the configuration engine, comprises an agent and middle to high level policy language for administering configuration tasks in large computer networks. Cfengine was originally designed to be a part of a "computer immune system". The name "autonomics" was coined much later by IBM.

Cfengine manages each device individually, placing all of the responsibility for policy conformance in the device itself. A single statement can result in many hundreds of links being created, or the permissions of many hundreds of files being set. The idea of cfengine is to create a single file or set of configuration files which will describe the setup of every host on your network. Cfengine runs on every host that reads a policy for configuration of the system; the configuration of the host is checked against this model and, if necessary, any deviations are fixed.

Cfengine approaches autonomics from the viewpoint of fixed-point behaviour – using a property that is usually called "convergence". Convergence is central to the design and functioning of cfengine. Cfengine's goal is to bring the system to a state of stable equilibrium and to thereafter maintain it in that state. Used appropriately, cfengine achieves this goal by carrying out a set of convergent operations derived from the statements in its configuration files and applied iteratively as required (recall that such an operation is one that can be repeated without further effect in/from the equilibrium state). Ref. [95] demonstrates that this approach is theoretically rigorous and (mathematically) well-defined.

Central to this approach is the notion that policies are applied locally within individual systems, with only emergent global consequences.

6.2.1 Context awareness in cfengine

Context can be thought of as an ontology of characteristics that describes the environmental conditions in which a computing device finds itself. Context can change

either because a device roams, or simply because the environment around it changes (e.g. in a data centre). Again, there are no issues of principle that are particular to wireless connectivity, as opposed to fixed-line connectivity.

Cfengine identifies its environment in three ways by measuring it and classifying it each time the agent is invoked.

- Detection of device capabilities and type (software probes).
- Detection of current configuration state (configuration probes).
- Detection of relative state of resource usage (machine learning).

A cfengine class is a boolean identifier which provides an ontological description of the state of the environment. It is a way of slicing up and mapping out the complex environment of one or more hosts into regions that can then be referred to by a symbol or name. A single class can represent relatively fixed aspects of the device, such as its operating system, current revision, the time of day and so on:

- The name of an operating system architecture e.g. `ultrix, solaris`, etc.
- The identity of a particular device.
- The name of a user-defined group of devices.
- A day of the week (in the form `Monday, Tuesday, Wednesday, ..`).
- An hour of the day (in the form `Hr00, Hr01 ... Hr23`).
- Minutes in the hour (in the form `Min00, Min17 ... Min45`).
- A day of the month (in the form `Day1 ... Day31`).
- A month (in the form `January, February ... December`).
- A year (in the form `Yr1997, Yr2004`).
- An arbitrary user-defined string.
- The IP address octets of any active interface (in the form `ipv4_192_0_0_1`, `ipv4_192_0_0`, `ipv4_192_0`, `ipv4_192`).

Cfengine also provides a number of built-in functions for evaluating the state of the device resources, in the context of the local operating system, based in specific policy-dependent tests, e.g.,

```
classes:
access_to_dir = ( ReturnsZero(/bin/cd /mydir) )
compare = ( ChangedBefore(/etc/passwd_master,/etc/passwd) )
isplain = ( IsPlain(/tmp/import) )
inrange = ( IPRange(128.39.89.10-15) )
CIDR = ( IPRange(128.39.89.10/24) )
compute_nodes = ( HostRange(cpu-,01-32)
gotinit = ( PrepModule(startup2,"arg1 arg2") )
```

Finally, other classes are automatically evaluated based on the state of the host, in relation to the learned condition of the system at earlier times. A database of system averages and variances, which characterize “normal” behaviour, is maintained. The state of the system is examined and compared to this database in real time, and the state is classified in terms of the current level of activity, as compared to an average of equivalent earlier times, e.g.

```
RootProcs_low_dev2
netbiosssn_in_low_dev2
smtp_out_high_anomalous
www_in_high_dev3
ftp_in_high_microanomaly
```

The first of these tells us that the number of root processes is two standard deviations below the average of past behaviour, which might be fortuitous, or might signify a problem, such as a crashed server. The WWW item tells us that the number of incoming connections is three standard deviations above average. This kind of self-analysis is only possible on an operating system with the sophistication of multitasking environment.

Cfengine organizes the information surrounding a resource condition first in terms of statistical significance and then only later in terms of event characteristics. Other criteria are needed to pin down which anomalies are interesting and which are not. As a second level of policy filtering, cfengine provides a measure of the entropy of the source IP addresses of the measured data. A low entropy value means that most of the events came from only a few (or one) IP addresses. A high entropy value implies that the events were spread over many IP sources. These conditions are described by classes of the form:

```
entropy_www_in_high
entropy_smtp_in_low
```

Thus, for example, in the first case the class will be set if incoming traffic at the peak event of the last data sample was spread evenly over all the incoming addresses. Such an event indicates that the resource usage is not due to a single source (e.g. an attacker from a single location) but is evenly spread — perhaps just a coincidental anomaly. In the second case, the low entropy SMTP traffic must come from one or two addresses and is more likely to be spam or an attack of some kind. These classes can be combined with the specific anomaly thresholds (see example below).

The resulting class list, obtained from exploring the environment of the system, and after parsing a configuration, looks something like this:

```
host% cfagent -p -v
[snip]
Defined Classes = ( any Thursday Hr14 Min24 Min20 25
Day19 July Yr2001 solaris examplehost 32_bit sunos 5 7
sunos_sun4u sunos_sun4u 5 7 sparc solaris2 7 129 0 0
129 0 0 10 loghost OnTheHour peaktime DayTime
examplehost example_org longjob Setup SSH OK y MailHub
percent 60 RootProcs normal dev2 nfsd out low dev2 )
```

6.2.2 Operator ordering

The order in which configuration operations are performed is an important aspect of detailed autonomic computability. If ordering is important, a system can get stuck in a bad state. Cfengine handles this problem in a pragmatic way, allowing it to operate in any scheduling scheme – based either on events or regular sweep scheduling. Discussions of ordering of operations have been presented in, for example, refs. [96],

[97]. The most important result from this on-going area of intensive research is that ordering of operations is not important provided that operations are orthogonal. When this is the case, simple iteration results in ordering conflict resolution provided that the individual operators are themselves convergent.

6.3 Autonomy and Voluntary Cooperation

Cfengine employs a security model that disallows external management of devices. It is purposely deaf to all external comments except one: the “wake up” instruction. Even then, this command is made available to external agents only with a limited access and frequency to avoid abuse (Denial of Service attacks).

Describing behaviour without external managers is unfamiliar in the literature. To describe the effects of these principles on management, a different kind of modelling language is required that can deal with autonomy and probability: promise theory.

6.3.1 Promise Theory

Promise theory was invented to discuss the issues surrounding autonomous operation, and voluntary cooperation. Unlike other modelling techniques, like Petri Nets or UML, promise theory is not about the stepwise development of a device. It is not about protocol modelling; rather it is about *equilibria*, i.e. how to describe steady state behaviour that has some underlying dynamics.

Promise theory begins with the idea of completely autonomous agents that interact through the promises they make to one another. It is therefore particularly well-suited to modelling cfengine.

Cfengine complies with this principle of host autonomy. Many existing systems and descriptions of configuration management assume a centralized authority to change all the parts of the system. This has a number of disadvantages. Our aim here is to begin in the opposite manner, assuming *no centralization* of authority, and then seeing how collaborative structures emerge by free choice.

The way to secure a clear and consistent picture of policy, in such complex environments, is through the use of formal methods. Cfengine’s policy language is very close to promise theory’s description.

6.3.2 Policy with autonomy

By a policy we mean the ability to assert arbitrary constraints of the behaviour of objects and agents in a system. The most general kind of system one can construct is a collection of objects, each with its own attributes, and each with its own policy. A policy can also be quite general: e.g. policy about behaviour, policy about configuration, or policy about interactions with others.

In a network of *autonomous* systems, an agent is only concerned with assertions about its own policy; no external agent can tell it what to do, without its consent. This is the crucial difference between autonomy and centralized management, and it will be the starting point here (imagine privately owned devices wandering around a shopping mall).

No agent can force any other agent to accept or transmit information, alter its state, or otherwise change its behaviour.

6.3.3 Cfengine statements are promises

A promise is a general and abstract unit of past, present or future behaviour. Promises, between agents, can deal with things like quality of service, quality of behaviour, specifications of state, etc. Policies of various types have been identified. For instance, in the Ponder model [98], one has authorizations (promises to grant access) and obligations (promises to follow up on a different promise) or dependency, etc. These can all be translated into the notion of promises by rethinking.

Consider, then, a set of autonomous agents of objects represented as nodes $N = \{n_1, n_2, ..., n_N\}$ in a graph. A promise is a labelled directed edge (link) that connects two nodes. The promise label represents a specifically intended range of behaviour χ from within a domain of possible behaviours. i.e. $n_1 \xrightarrow{\chi} n_2$. A promise is thus made by a node n_1 to a node n_2. A promise is assumed to be always kept.

Example 1 (Service Level Promise) *Agent* n_1 *promises agent* n_2 *to provide service of type 'database access in time* q*'.* τ *is the type domain* $q \in [0, \infty)$ *and the constraint* $\chi(q): 0 < q < 10ms.$

The formulation of a promise, above, has obvious characteristics of a directed graph. In this elementary scheme, the only inconsistency that can occur is if an agent promised two contradictory things to a second agent. A promise of χ_1 and type τ from agent n_1 to agent n_2 is said to be broken if there exists another promise from n_1 to n_2, of χ_2 and type τ, in which $\chi_1 \neq \chi_2$.

According to this prescription, an agent can only break its own promises: if an agent promises two different things, it has broken both of its promises. This makes the logic of promises considerably simpler than the logic of obligation. Two promises of different types do not contradict one another, but two promises about the same thing are clearly inconsistent. Put another way, the intersection of any χ_1, χ_2 of the same type must be the null set for all *i* and *j*.

6.3.4 Uniformity through collaboration

In a managed network, one often strives for large scale consistency. How can this be reconciled with a view in which every node is independent? What is required is a uniformity based on freely given promises that allows independent agents to come together and form structures spanning several nodes; i.e. by *voluntary cooperation*. Such an agreement has to be made between every pair of nodes involved in the cooperative structure.

6.4 Introduction to Distributed Management Patterns

The Pattern-based management paradigm is a distributed management method based on the use of graph traversal algorithms to control and coordinate the processing and aggregation of management information inside the network. From the perspective of a network manager, the algorithm provides the means to diffuse or spread the computational process over a large set of nodes.

The paradigm achieves this through the development of two important concepts; the navigation pattern and the aggregator. The former represents the generic graph traversal algorithms that implement distributed control while the latter implements the computations required realizing the task. A navigation pattern controls the flow of

execution of a (distributed) management operation. It is described by an asynchronous network algorithm, which can be analyzed for its complexity and scalability properties.

Pattern-based management aims at overcoming the limitations of centralized management: its poor scalability regarding a) management traffic; b) processing load on the management station, and c) execution times. Its goal is to build scalable, robust and adaptable management systems. The main benefits of Pattern-based management are that (i) separates the semantics of the task from its flow control, (ii) enables building scalable management systems, (iii) facilitates management in dynamic environments, and (iv) does not require a priori knowledge of the network topology, in contrast to centralized approaches. Figure 6-1 presents the simplest examples of navigation patterns.

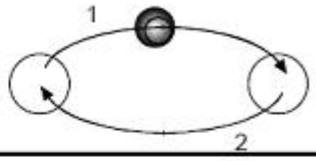
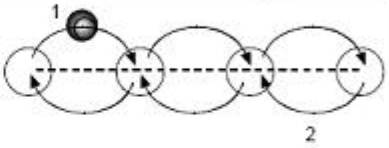
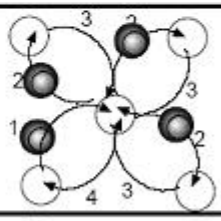
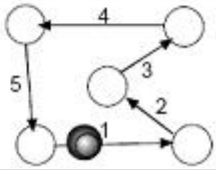
operation	typical application	navigation pattern
type 1: node-to-node	1 node control/monitor (get/set of variables)	
type 2: visit all nodes along a path/flow	1 flow/path control e.g.: traceroute, bottleneck detection, signalling, VPN operation	
type 3: distribute agent to all nodes in subnet (parallel control)	subnet control, message broadcast e.g.: congestion location detection	
type 4: visit all nodes in subnet (sequential control)	subnet control e.g.: topology detection	

Figure 6- 1 Example navigation patterns

Pattern-based applications map network-wide operations (such as identifying the top flows regarding traffic) into local operations that will be performed by the managed nodes. Then, they are distributed using navigation patterns, creating an execution graph. For monitoring tasks, local operations include the collection of statistics and the incremental aggregation of the collected data. This aggregation is done in parallel across the network asynchronously: all the nodes contribute in the calculation.

Navigation patterns have been shown to improve the scalability of network management and monitoring operations by decentralizing and locally aggregating computational and network resource usage in large-scale network environments [99] [91]. We believe these benefits, which can be measured in terms of time and traffic complexity [90], are also portable to environments where the object is a network of multi-user dynamic systems.

6.4.1 Navigation patterns for distributed autonomics

We can extend the idea of autonomic behaviour by allowing a distributed cluster of devices to monitor themselves as a group and exchange certain information about the aggregate group in an efficient manner.

Navigation patterns are characterized by the messaging flow imposed by the combination of an overlay network and communication policies as well as the usage of computing power throughout the network to act on data (as opposed to all calculation occurring in a single node). The product of navigation patterns is a capability to perform management operations on a network and have both the link usage and the computational intensity distributed throughout the network rather than focused at a single node. A brief summary of recently defined navigation patterns follows.

6.4.2 Echo

A simple example is the *echo* pattern. The fundamental topology of the echo pattern is a spanning tree, with a very central node in the network as the root of the tree. The pattern has two phases of communication: *expansion* and *contraction*. During the expansion phase, the root node issues a query to its children. Each node in the tree repeats this process. The contraction begins as the query reaches a leaf node. The leaf node answers the query, sending its response to its parent in the tree. The parent receives the response of its children, aggregates or calculates information for the query to the fullest extent possible, and then sends a single aggregate answer to its own parent. This process is repeated back to the root node, which finally receives and aggregates the messages from its children. The tree topology provides for parallelized execution while the aggregation of query responses during contraction reduces the amount of traffic that would otherwise be necessary [90].

6.4.3 Generic Aggregation Protocol

In contrast to the echo pattern in which the managing node queries other nodes in the network, the Generic Aggregation Protocol (GAP) employs a strategy where the managing node is query-passive and receives continuous updates from other nodes in the network. GAP creates a self-stabilizing, breadth-first spanning tree that accommodates for network and node failures by maintaining neighbourhood information at each node. The ways in which the neighbourhood table at a node changes in response to changes in its neighbours can be modified by policy, which allows for general policy-level adjustment of how the network automatically forms and stabilizes [91].

6.4.4 TCA-GAP

An extension of GAP, TCA-GAP is concerned with aggregated threshold crossing alerts. This method allows for the reduction of total traffic complexity in a network by restricting the number of updates sent to the parents (in terms of the overlay network tree) of the region for which a threshold is defined. The local thresholds are derived from a global threshold and negotiated by neighbours in the overlay. This distributed approach stands in contrast to the typical current practice of monitoring thresholds at only the network edge and processing that information on a managing node, which is often dedicated to the task due to its computational intensity [100].

6.4.5 A-GAP

Another extension of GAP, A-GAP addresses the challenge of specifying a monitoring accuracy level while minimizing the necessary management overhead. Typical rate-control methods reduce traffic complexity but without regard to the impact on accuracy. A-GAP, on the other hand, allows for the specification of an accuracy threshold, below which local filters will prevent updates from being pushed to the managing node. The global accuracy threshold is first specified on the managing node then distributed and re-computed throughout the network as local filters [101].

6.5 Integration work

By providing support for pattern-based communication, cfengine can be made optimal with respect to distributed information exchange. A single pattern-based exchange can sample the entire local environment of a device by aggregating state from its neighbourhood. This has the potential to enable the idea of distributed autonomics without sacrificing autonomy or partial connectivity (ubiquitous computing).

Cfengine presently supports a configurable peering model. However, the software's flexibility allows policy to invoke cfengine with various forms of machine-readable information and act on it. Thus one could configure an implicit overlay network necessary to implement the navigation patterns.

Cfagent is the principal component of cfengine; it will both copy configuration files from a master policy host as well as execute a policy as described in a configuration file on the local host. Cfservd is the daemon, which both listens for requests to execute cfagent and serves file copies. Cfrun is a command which can be used to request the execution of cfagent on another host. Each of these components was used in the experiment described below.

6.5.1 Basic testing

The first step of the integration has been to implement the simplest navigation pattern (echo) without modification to cfengine. This is possible entirely using the basic policy framework.

```
control:
actionsequence = ( editfiles shellcommands )
domain = ( echonet )
workdir = ( /tmp )
children = ( ReadList("$(workdir)/hosts.$(host)",lines,#,1000) )
editfiles:
/etc/hosts
BeginGroupIfFileExists "$(workdir)/hosts.$(host)"
DefineInGroup "HasChildren"
EndGroup
shellcommands:
any::
"/bin/echo Pattern reached device $(host)"
HasChildren::
"/usr/local/sbin/cfrun $(children)"
```

The initial test setup includes a network of 6 Xen virtual hosts running a full installation of Ubuntu GNU/Linux (Dapper) Server; 6 hosts are sufficient for demonstrating a very simple spanning tree. The virtual hosts were running on a single physical host, bridged together on the same virtual LAN (Figure 6-2).

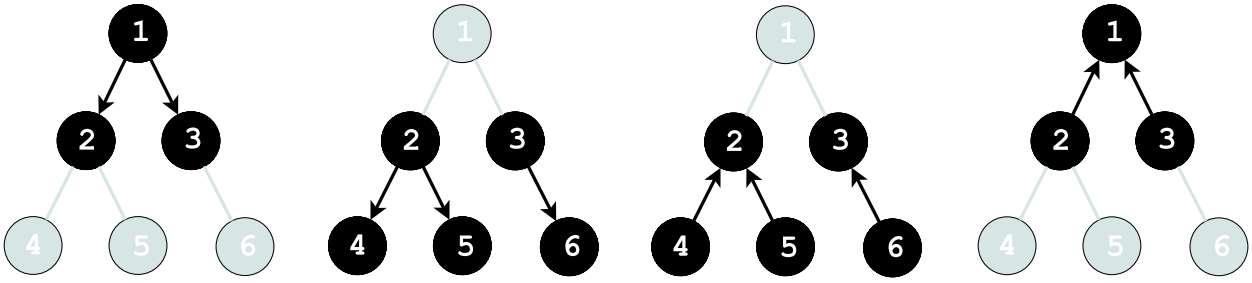


Figure 6-2 Step-by-step echo pattern communication behaviour for the 6-node network in initial tests

While cfengine allows the possibility to use a dynamic network overlay in user-space, the simplest first step for the small 6-node network was to manually tell each node about its children. This was done by creating a host file (named `hosts.$node`) for each node in the network that was not a leaf. The cfagent policy checked for the existence of that file; if it existed, cfrun was executed for each child in the list. In this way, the spanning tree overlay was formed. Cfagent's natural behaviour is to return the aggregate output of any commands it spawns, with cfrun as no exception. And so, cfengine naturally fulfilled the aggregation requirement of the echo pattern.

The test consisted of a simple command for each node to print its hostname. It was executed 20 times in a serialized star topology, and 20 times with the echo pattern.

6.5.2 Initial results

Our initial results on an admittedly small network of hosts indicate a slightly smaller average execution time for the echo pattern compared to the star pattern; see the table in Figure 6-3.

Pattern	Exec Time
Star	1.6674
Echo	1.6265

Figure 6-3 Star and echo pattern test results for 6 nodes

6.5.3 Future work and applicability

Work to this point is merely a proof of concept. The laboratory setup is limited in several key ways, which are planned to be resolved in the next year:

- The test network is currently very small. It needs to be scaled up significantly for more meaningful analysis and be repeated over statistically valid samples.
- The test network is currently a set of virtual machines on a single host. Therefore messages sent in close time proximity may compete for resources in setups with a high volume of nodes. While navigation patterns serve to minimize traffic complexity of monitoring and management operations, this matter deserves investigation.
- The echo pattern in cfengine has presently been accomplished through manual configuration of host list files. In order to be practical and scalable, cfengine's implementation of navigation patterns needs to be dynamic, therefore

accommodating for node failures and topology changes (which means implementing GAP). GAP-based protocols do not specify a method for neighbour discovery and failure detection [91]. Ref [102] shows a Java service using Delaunay triangulation to create overlay networks used for the echo pattern, GAP, and TCA-GAP.

Additionally, we plan to consider the following factors:

- How does the use of navigation patterns compare with random scheduling?
- Cfengine supports a variety of operations encompassing monitoring and management; how do navigation patterns perform on different types of operators (e.g. monitoring, configuration actions, file distribution)?
- How does the use of navigation patterns on cfengine-managed systems impact the cooperation dynamics of those systems?
- Does the work on navigation patterns inspire other event-driven approaches that can further reduce the time or traffic complexity of management or monitoring actions?

7 Conclusions

We have studied and reviewed fundamental approaches and frameworks towards the autonomic management of fixed IP QoS-enabled networks and ubiquitous systems, with emphasis being placed on ad hoc networks. The variety of frameworks and approaches that was exhibited through our work in this deliverable, the various aspects of autonomic management, in the sense of self-* properties, were addressed. Our future work relies on building upon these frameworks to establish viable and practical solutions for the autonomic management of networks in an integrated, unified manner. We also plan to study technologies that act as enablers of autonomic management and these will be the focus of the upcoming deliverable D9.2.

We presented an overview on descriptions regarding the context-related research activity for this deliverable; we have identified major problems and their possible solutions, as well as basic concepts pertinent to the context information in autonomic communications systems as likewise the required functionality and interactions for provisioning such services. We have furthermore provided pointers for open issues and general conclusions in the context-awareness and self-awareness realm in regard to autonomic systems.

The management of ubiquitous environments is definitely harder and more complex than the management of traditional fixed environments. Indeed, ubiquitous environments are highly dynamic and are typically self-maintained by mobile devices that operate under many constraints such as bandwidth limitation, energy exhaustion, and low system capacities. They impose new challenges on any attempt to effectively manage them. Autonomic management is a promising area of research in this context and covers the aforementioned functional areas of self-configuration, self-healing, self-optimization and self-protection. The objective is to model, design and experiment new management architectures and protocols to provide to these environments the capabilities to manage themselves and to overcome their changes in a dynamic and autonomous manner. In this light, we have provided a comprehensive study on self-management issues for ubiquitous environments.

A proof of concept approach to distributed autonomics, built on the foundations outlined by the previously mentioned studies concluded this deliverable. More specifically, cfengine is an established framework for autonomic computing, used on up to a million computers around the world. Management patterns are an efficient way of signalling in overlay networks enabling distributed collaboration. The aim of the particular part of the deliverable work was to allow cfengine to benefit from management pattern technology without sacrificing its basic principles of autonomy. Experimental results provided proof for successful deployment.

Successful collaboration links among all partners of WP9 were built and the work presented in this deliverable provides evidence for this. Integration work and common research activities were performed and planning for future joint research actions has been streamlined.

8 References

[1] Haas, R., Droz, P. and Stiller, B., “Autonomic service deployment in networks”, IBM Systems Journal, Vol. 42, No 1, 2003

[2] Kephart, J.O. and Chess, D.M., “The Vision of Autonomic Computing”, IEEE Computer, January 2003

[3] Ganek, A. G. and Corbi, T.A., “The dawning of the autonomic computing era”, IBM Systems Journal, Vol. 42, No 1, 2003

[4] Dey, A. K., Understanding and using context, Journal of Personal and Ubiquitous Computing, Volume 5 (1), pp. 4-7, 2001

[5] Strassner, J. and Kephart, J., Autonomic Networks and Systems: Theory and Practice, Network Operation Management Symposium. NOMS 06 Tutorial, April 2006.

[6] Serrano, J.M., Serrat, J. and O'Sullivan, D.; "Onto-Context Manager Elements Supporting Autonomic Systems: Basis & Approach" . 2006 IEEE ManWeek in 1st IEEE MACE: Modelling Autonomic Communications Environments. 23-27 October 2006, Dublin, Ireland.

[7] Serrano, J.M., Serrat, J., Yang, K., Salamanca, E.; "Modelling Context Information for Managing Pervasive Network Services". 2005 Proceedings of International Conference on Modelling and Simulation - ICMS' 05, AMSE/IEEE Morocco Section, 22 - 24 November 2005, Marrakech, Morocco.

[8] Abowd, G. D., Ebling, M., Hunt, G., Lei, H., Gellersen, H.-W., Context-Aware Computing, Guest Editorial, IEEE Pervasive Computing, July-September 2002

[9] Satyanarayanan, M., Pervasive Computing: Vision and Challenges, IEEE Personal Communications, August 2001

[10] Weiser, M., The Computer for the 21st Century, Scientific American, September 1991

[11] Fang, Y., McDonald, A.B., Cross-layer performance effects of path coupling in wireless ad hoc networks: power and throughput implications of IEEE 802.11 MAC, in Proc. 21st IEEE International Performance, Computing, and Communications Conference, pp. 281-290, April 2002

[12] Gray, P.D., Salber, D., Modelling and Using Sensed Context Information in the Design of Interactive Applications, in Proc. 8th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI'01), May 2001

[13] Mendes, P., Prehofer, C., Wei, Q., Context management with programmable mobile networks, IEEE Computer Communication Workshop, 2003

[14] Pascoe, J., N. Ryan, N., Morse, D., Issues in developing context-aware computing, In Proc. First International Symposium on Handheld and Ubiquitous Computing (HUC'99), 1999

[15] Dey, A.K., Abowd, G.D., Towards a better understanding of context and context awareness, in Workshop on the What, Who, Where, When and How of Context-Awareness, affiliated with the 2000 ACM Conference on Human Factors in Computer Systems (CHI 2000), April 2000

[16] Malatras, A., Pavlou, G., Gouveris, S., Sivavakeesar, S. Karakoidas, V., Self-Configuring and Optimizing Mobile Ad Hoc Networks, IEEE International Conference on Autonomic Computing (ICAC 05), 2005

[17] Hadjiantonis, A., Malatras, A., Pavlou, G., A context-aware, policy-based framework for the management of MANETs, IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 06), 2006

[18] Malatras, A., Hadjiantonis, A., Pavlou, G., Exploiting context-awareness for the autonomic management of mobile ad hoc networks, to appear in Journal of Network and Systems Management, Special Issue on Autonomic, Pervasive and Context-Aware Systems, March 2007

[19] A. Malatras and G. Pavlou, Context-driven Self-Configuration of Mobile Ad hoc Networks, IFIP International Workshop on Autonomic Communications (WAC 2005), October 2005

[20] Malatras, A., Gouveris, S., Sivavakeesar, S., Pavlou, G., Programmable, Context-Aware Middleware for the dynamic Deployment of Services and Protocols in Ad Hoc Networks, 12th Annual HP-OVUA Workshop, 2005

[21] Henricksen, K., Indulska, J., and Rakotonirainy, A. Generating Context Management Infrastructure from High-Level Context Models. In Industrial Track Proceedings of the 4th International Conference on Mobile Data Management (MDM2003) (Melbourne/Australia, January 2003), pp. 1-6

[22] Henricksen, K., Indulska, J., and Rakotomirainy, A. Modeling context information in pervasive computing systems. In LNCS 2414: Proceedings of 1st International Conference on Pervasive Computing (Zurich, Switzerland, 2002), F. Mattern and M. Naghshineh, Eds., Lecture Notes in Computer Science (LNCS), Springer, pp. 167-180.

[23] Serrano, J.M., Serrat, J.; "Context Modelling and Handling In Context-Aware Multimedia Applications". IEEE Wireless Communications Magazine 2006, Special Issue on Multimedia in Wireless/Mobile Ad-hoc Networks, October 2006.

[24] Schilit, B. N., Adams, N. L., and Want, R. Context-aware computing applications. In IEEE Workshop on Mobile Computing Systems and Applications (Santa Cruz, CA, US, 1994).

[25] Held, A., Buchholz, S., and Schill, A. Modeling of context information for pervasive computing applications. In Proceedings of SCI 2002/ISAS 2002 (2002).

[26] ESPIRIT PROJECT 26900: Technology for enabled awareness (tea), 1998.

[27] GUIDE Project: Understanding Daily Life via Auto-Identification and Statistics http://seattleweb.intel-research.net/projects/guide/

[28] McCarthy, J. Notes on formalizing contexts. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (San Mateo, California, 1993), R. Bajcsy, Ed., Morgan Kaufmann, pp. 555-560.

[29] Ötzürk, P., and Aamodt, A. Towards a model of context for case-based diagnostic problem solving. In Context-97; Proceedings of the interdisciplinary conference on modeling and using context (Rio de Janeiro, February 1997), pp. 198-208.

[30] Chen, H., Finin, T., and Joshi, A. Using OWL in a Pervasive Computing Broker. In Proceedings of Workshop on Ontologies in Open Agent Systems (AAMAS 2003) (2003).

[31] H. Chen, T. Finin, and A. Joshi. An Ontology for Context-Aware Pervasive Computing Environments. In IJCAI workshop on ontologies and distributed systems, IJCAI'03, August, 2003.

[32] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", December 1998, Internet RFC 2475.

[33] www.ist-tequila.org

[34] V. Sarangan and J. Chen, "Comparative Study of Protocols for Dynamic Service Negotiation in Next Generation Internet", IEEE Communications Magazine, March 2006.

[35] Rashid J. Al-Ali, Omer F. Rana, David W. Walker, Sanjay Jha, and Shaleeza Sohaol. G-QoSM: Grid Service Discovery Using QoS Properties. COMPUTING AND INFORMA TICS, 2002.

[36] Juan I. Asensio and Villagr Vtor A. A UML Profile for QoS Management Information Specification in Distributed Object-based Applications. In 7th International Workshop of the HP Open View University Association (HPOVUA 2000), Santorini, Greece, June 2000.

[37] Juan I. Asensio, Villagr Vtor A., Jorge E. Lez-de Vergara, and Julio J. Berrocal. UML Profiles for the Specification and Instrumentation of QoS Management Information in Distributed Object-based Applications. In 5th Fifth World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001), Orlando, Florida, July 2001. ISBN 980-07-7543-9.

[38] C. Becker and K. Geihs. Generic QoS Specifications for CORBA. In Proceedings of Kommunikation in verteilten Systemen (KIVS'99), Darmstadt, 1999.

[39] G. Dreo Rodosek. Quality Aspects in IT Service Management. In M. Feridun, P. Kropf, and G. Babin, editors, Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2002), Lecture Notes in Computer Science (LNCS) 2506, pages 82–93, Montreal, Canada, October 2002. IFIP/IEEE, Springer.

[40] E. Exposito, M. Gineste, R. Peyrichou, P. Senac, and M. Diaz. XQOS : XML-based QoS specification language. In Proceedings of 9th International Conference on Multi-Media Modeling, Taiwan, January 2003.

[41] Patria Gomes Soares Florissi. QoSME: QoS Management Environment. Phd thesis, Columbia University, 1996.

[42] Svend Frlund and Jari Koistinen. Qml: A language for quality of service specification. Report hpl-98-10, Software Technology Laboratory, Hewlett-Packard Company, September 1998.

[43] M. Garschhammer, R. Hauck, H.-G. Hegering, B. Kempter, I. Radisic, H. Roelle, and H. Schmidt. A Case–Driven Methodology for Applying the MNM Service Model. In R. Stadler and M. Ulema, editors, Proceedings of the 8th International IFIP/IEEE Network Operations and Management Symposium (NOMS 2002), pages 697–710, Florence, Italy, April 2002. IFIP/IEEE, IEEE Publishing.

[44] M. Garschhammer, R. Hauck, B. Kempter, I. Radisic, H. Roelle, and H. Schmidt. The MNM Service Model — Refined Views on Generic Service Management. Journal of Communications and Networks, 3(4): 297–306, December 2001.

[45] Markus Garschhammer and Harald Roelle. Requirements on quality specification posed by service orientation. In Proceedings of the 15th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2004), pages 1–14, 2004 2004.

[46] Xiaohui Gu, Duangdao Wichadakul, and Klara Nahrstedt. Visual QoS Programming Environment for Ubiquitous Multimedia Services. In Proceedings of IEEE International Conference on Multimedia and Expo 2001 (ICME2001), Tokyo, Japan, August 2001

[47] Information Processing Systems – Open Systems Interconnection – Basic Reference Model. IS 7498, ISO/IEC, 1984

[48] J.P. Loyall, D.E. Bakken, R.E. Schantz, J.A. Zinky, D.A. Karr, R. Vanegas, and K.R. Anderson. QoS Aspect Languages and Their Runtime Integration. In Lecture Notes in Computer Science, Vol. 1511, Proceedings of the Fourth Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers (LCR98), Pittsburgh, Pennsylvania, May 1998. Springer-Verlag.

[49] TINA Object Definition Language Manual. Technical report, TINA-C, July 1996.

[50] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. RFC 2330: Framework for ip performance metrics. RFC, IETF, May 1998.

[51] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller. A Concept for QoS Integration in Web Services. In Proceedings of 1st Web Services Quality Workshop (WQW 2003), in conjunction with 4th International Conference on Web Information Systems Engineering (WISE 2003), December 2003.

[52] Badonnel, R., State, R., Festor, O.: Probabilistic Management of Ad-Hoc Networks. In: Proc. of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS'06), Vancouver, Canada (2006)

[53] Burgess, M., Canright, G.: Scalability of Peer Configuration Management in Logically Ad-hoc Networks. IEEE eTransactions on Network and Service Management (eTNSM) 1(1) (2004)

[54] D'Souza, R., Ramanathan, S., Land, D.T.: Measuring Performance of Ad-hoc Networks using Timescales for Information Flow. In: Proc. of IEEE International Conference on Computer Communications (INFOCOM'03), San Francisco, CA, USA (2003)

[55] Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). http://www.ietf.org/rfc/rfc3626.txt (2003) IETF RFC 3626.

[56] Lymberopoulos L., E. Lupu and M. Sloman. An Adaptive Policy Based Framework for Network Services Management, Plenum Press Journal of Network and Systems Management, Special Issue on Policy Based Management, 11: 3 Sep. 2003, pp277-303

[57] SAMAN: NS-2 Network Simulator. http://www.isi.edu/nsnam/ns/ (1989)

[58] Universal Plug and Play Device Architecture. http://www.upnp.org/resources/documents.asp

[59] Yoon, J., Liu, M., Noble, B.: Random Waypoint Considered Harmful. In: Proc. of IEEE International Conference on Computer Communications (INFOCOM'03), San Francisco, CA, USA (2003) 1312–1321

[60] Boudec, J.Y.L., Vojnovic, M.: Perfect Simulation and Stationarity of a Class of Mobility Models. In: Proc. of IEEE International Conference on Computer Communications (INFO- COM'05), Miami, FL, USA (2005)Autonomic Computing Special Issue, IBM Systems Journal, Vol. 42, No 1, 2003.

[61] HP Utility Data Center: Enabling Enhanced Datacenter Agility, http://www.hp.com/large/globalsolutions/ae/pdfs/udc_enabling.pdf, May 2003

[62] Elvin http://www.elvin.org/

[63] xmlBlaster http://www.xmlblaster.org/

[64] Siena Wide Area Event Notification Content Based Routing http://serl.cs.colorado.edu/~serl/ dot/siena.html

[65] G. Pavlou, P. Flegkas, S. Gouveris, A. Liotta, On Management Technologies and the Potential of Web Services, IEEE Communications, special issue on XML-based Management of Networks and Services, Vol. 42, No. 7, pp. 58-66, IEEE, July 2004.

[66] Wahl, M., T. Howes and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, Dec. 1997

[67] Ponder2 http://www.ponder2.net

[68] Moore, B., Elleson, E., Strassner J., Westerinen, A, "Policy Core Information Model-Version 1 Specification", RFC 3060, Feb.2001

[69] Moore B., "Policy Core Information Model (PCIM) Extensions", RFC 3460, Jan.2003

[70] Strassner J., Moore, B., Moats, R., Elleson, E., "Policy Core Lightweight Directory Access Protocol (LDAP) Schema", RFC3703, Feb.2004

[71] Pana M., Reyes, A. Barba, A., Moron, D., Brunner, M., "Policy Core Extension Lightweight Directory Access Protocol Schema (PCELS)", RFC 4104, Jun.2005

[72] D.Mandato, E.Kovacs, F.Hohl, and H.A-Alikhani, CAMP: A context-aware mobile portal, IEEE Communications Magazine, no. 1, Jan. 2002, pp. 90-97.

[73] B.N.Schilit, D.M.Hilbert, and J.Trevor, "Context-aware communication", IEEE Wireless Communications, no. 5, Oct. 2002, pp. 46-54.

[74] XML-RPC specifications web site, http://www.xmlrpc.com/spec.

[75] Badonnel, R., State, R., Festor, O.: Fault Monitoring in Ad-Hoc Networks Based on In- formation Theory. In: Proc. of the 5th International IFIP-TC6 Networking Conference (NETWORKING'06), Coimbra, Portugal, Lecture Notes in Computer Science 3976, Springer Verlag (2006)

[76] Kherani, A., Altman, E., Michiardi, P., Molva, R.: Non-cooperative Forwarding in Ad- hoc Networks. In: Proc. of the International IFIP Networking Conference (Networking'05), Waterloo, Canada (2005)

[77] Shannon, C.E.: A Mathematical Theory of Communication. The Bell System Technical Journal 27 (1948) 379–423

[78] Jacquet, P., Szpankowski, W.: Entropy Calculation via Analytic Depoissonization. IEEE Transaction on Information Theory 45 (1999) 1072–1081

[79] Westphal, C.: On Maximizing the Lifetime of Distributed Information in Ad-Hoc Networks with Individual Constraints. In: Proc. of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'05), Urbana-Champaign, IL, USA (2005)

[80] Zweig, M., Campbell, G.: Receiver-Operating Characteristic (ROC) Plots: a Fundamental Evaluation Tool. Clinical Chemistry 29(4) (1993) 561–577
[81] Jakobson, G., Weissman, M.D.: Real-time Telecommunication Network Management: Extending Event Correlation with Temporal Constraints. In: Proc. of the 4th IFIP/IEEE International Symposium on Integrated Network Management (IM'95), Santa Barbara, CA, USA (1995)
[82] Zhuang, S.Q., Geels, D., Stoica, I., Katz, R.H.: On Failure Detection Algorithms in Over- lay Networks. In: Proc. of IEEE International Conference on Computer Communications (INFOCOM'05), Miami, FL, USA (2005)
[83] Baccelli, E., Rajan, R.: Real-Time OSPF Route Monitoring. In: Proc. of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM'01), Seattle, WA, USA (2001)
[84] Ramachandran, K., Belding-Royer, E., Almeroth, K.: DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks. In: Proc. of IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04), Santa Clara, CA, USA (2004)
[85] Ngo, D., Wu, J.: WANMON: a Resource Usage Monitoring Tool for Ad-hoc Wireless Net- works. In: Proc. of the 28th Annual IEEE Conference on Local Computer Networks (LCN'03), Bonn, Germany (2003) 738–745
[86] Badonnel, R., State, R., Festor, O.: Management of Mobile Ad-Hoc Networks : Evaluating the Network Behavior. In: Proc. of the 9th IFIP/IEEE International Symposium on Integrated Network Management (IM'05), Nice, France (2005) 17–30
[87] Badonnel, R., State, R., Festor, O.: Management of Mobile Ad-Hoc Networks: Information Model and Probe-based Architecture. ACM International Journal of Network Management (ACM IJNM) 15(5) (2005)
[88] M. Burgess. Cfengine www site. http://www.cfengine.org, 1993.
[89] M. Burgess. A site configuration engine. Computing systems (MIT Press: Cambridge MA), 8:309, 1995.
[90] K.S. Lim and R. Stadler. A Navigation Pattern for Scalable Internet Management. In Proceedings of the 7th IFIP/IEEE Symposium on Integrated Network Management, May 2001.
[91] M. Damand R. Stadler. A generic protocol for network state aggregation. Radiovetenskapoch Kommunication (RVK), June 2005.
[92] M. Burgess and G. Canright. Scalability of peer configuration management in partially reliable and ad hoc networks. Proceedings of the VIII IFIP/IEEE IM conference on network management, page 293, 2003.
[93] R. Badonnel, R. State, and O. Festor. Management of mobile ad-hoc networks: evaluating the network behavior. In Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management, pages 17–30. IEEE Press, 2005.
[94] R. Badonnel, R. State, and O. Festor. Probabilistic management of ad-hoc networks. In Proceedings of 10th IFIP/IEEE Network Operations and Management Symposium NOMS 2006.
[95] M. Burgess. Configurable immunity for evolving human-computer systems. Science of Computer Programming, 51:197, 2004.
[96] S. Traugott. Why order matters: Turing equivalence in automated systems administration. Proceedings of the Sixteenth Systems Administration Conference (LISA XVI) (USENIX Association: Berkeley, CA), page 99, 2002.
[97] M. Burgess. Configurable immunity model of evolving configuration management. Science of Computer Programming, 51:197, 2004.

[98] N. Damianou, N. Dulay, E.C. Lupu, and M. Sloman. Ponder: a language for specifying security and management policies for distributed systems. Imperial College Research Report DoC 2000/1, 2000.

[99] K. Lim, R. Adam, and R. Stadler. Decentralizing Network Management. IEEE electronic Transactions on Network and Service Management (eTNSM), 1(2), 2004.

[100] F.Wuhib,M. Dam, R. Stadler, and A. Clemm. Decentralized Computation of Threshold Crossing Alerts. In 16th Workshop on Distributed Systems: Operations and Management (DSOM).

[101] A. Gonzalez Prieto and R. Stadler. Distributable Real-Time Monitoring with Accuracy Objectives. Technical report, KTH Royal Institute of Technology, Sweden, 2005.

[102] P. Hefti. A Simple Overlay Platform for Decentralized Monitoring. Master's thesis, KTH Royal Institute of Technology, Sweden, 2006.

9 Abbreviations

ANMP	Ad hoc Network Management Protocol
AODV	Ad Hoc on Demand Distance Vector
BE	Best Effort
CAS	Context Aware Service
CC	Connected Component
CCP	Context Collection Point
CDC	Connected Device Configuration
CDP	Context Decision Point
CF	Capability Function
CH	Cluster Head
CLDC	Connected Limited Device Configuration
CM	Cluster Manager
CMT	Context Management Tool
CN	Cluster Node
CR	Context Repository
CSCP	Comprehensive Structured Context Profile
DMTF	Distributed Management Task Force
DPR	Distributed Policy Repository
DTD	Document Type Definition
ECA	Event Condition Action
ER	Entity Relationship Model
GAP	Generic Aggregation Protocol
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
LDAP	Lightweight Directory Access Protocol
MANET	Mobile Ad Hoc Network
MN	Manager Node
MPR	Multi-Point Relay
OLSR	Optimized Link State Routing
ORM	Object-Role Modelling
PBNM	Policy-Based Network Management
PCIM	Policy Core Information Model

PCIMe	Policy Core Information Model Extensions
PDA	Personal Digital Assistant
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PMT	Policy Management Tool
PR	Policy Repository
QoS	Quality of Service
QoSSL	Quality of Service Specification Language
ROC	Receiver Operating Characteristic
RPC	Remote Procedure Call
RWP	Random Waypoint
SGML	Standard Generic Markup Language
SLA	Service Level Agreement
SLS	Service Level Specification
SMC	Self-Management Cell
SNMP	Simple Network Management Protocol
SrNP	Service Negotiation Protocol
TC	Topology Control
TN	Terminal Node
UCS	Ubiquitous Computing System
UML	Unified Modelling Language
UPnP	Universal Plug and Play
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language

10 Acknowledgements

This deliverable was made possible due to the large and open help of the WP9 team of the EMANICS consortium within the Network of Excellence, which includes all of the deliverable authors as indicated in the document control. Many thanks are owed to all.